Simulation-based optimization for product and process design

Driessen, L.

Publication date:
2006

Document Version
Publisher's PDF, also known as Version of record

Simulation-based Optimization for Product and Process Design

Simulation-based Optimization for Product and Process Design

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit van Tilburg, op gezag van de rector magnificus, prof. dr. F. A. van der Duyn Schouten, in het openbaar te verdedigen ten overstaan van een door het college voor promoties aangewezen commissie in de aula van de Universiteit op vrijdag 28 april 2006 om 14.15 uur door

Lonneke Theodora Driessen

geboren op 4 juni 1978 te Tegelen.

Promotor: prof. dr. ir. D. den Hertog
Copromotor: dr. H. J. M. Hamers

Acknowledgements

A journey has almost come to an end and I would like to take the opportunity to express my gratitude to the people who travelled along. If they had not accompanied me, I would certainly not have reached this destination.

First of all I want to thank the project team, Dick den Hertog (my promotor), Herbert Hamers (my co-promotor), and Ruud Brekelmans. We became a team for the research project on the SEQUEM toolbox five years ago, and that was the start of a fruitful cooperation. I enjoyed the many project meetings we had. They always replenished my energy, which was certainly needed every now and then. I will never forget how three of us could discuss a certain research issue for an hour, until Ruud would finally interrupt with a single phrase showing us that we were thinking in the wrong direction. I also value the other discussions we had together a lot (about career, religion, 'veertjes and dempertjes', and the always returning threats about my laudatio).

I would also like to thank my former employer CQM B.V. in Eindhoven for having offered me the opportunity to carry out this research in addition to my regular work as consultant. I could spend one day a week at Tilburg University and I was always involved in consultancy projects related to my research area. This of course, helped me a lot.

Many thanks go to my former colleagues Erwin Stinstra and Peter Stehouwer. We spent a lot of time together pondering over Design Optimization Techniques and applying them in optimization projects. I am sure that ideas that came up during those sessions ended up somewhere in this thesis. Erwin and I shared an office at Tilburg University: three o'clock, chocolate or ice-cream time, finally! It was fun.

Special thanks go to Marloes Gerichhausen, who assisted me in my last research by completing a master thesis on the subject of an objective-driven method based on global approximation functions for integer optimization problems. I enjoyed the period of working together with her and I am sure we both learned a lot from it.

I am grateful to Emile Aarts, Ruud Brekelmans, Tom Dhaene, Fred van Keulen, and Jack Kleijnen for joining Dick and Herbert in my thesis committee. Furthermore, I would like to acknowledge the financial support from EIT-io for two of our research projects.

The department of Econometrics and Operations Research I would like to thank for hosting me with hospitality, and for the nice departmental trips. Special thanks go to Annemiek Dankers for the administrative support during these five years, to Kuno Huisman for providing me with a LaTeX framework for my thesis, and to Hendri Adriaens for the LaTeX support during the final stages of writing.

Lastly, I want to thank my friends and family for all the fun and enjoyable events that kept me concentrated on work during the remaining days. Rik and Joep, it will feel good to have you near as my 'paranymfen' during the defence of my thesis. My parents I thank for always being there for me. I can always count on you! And Ulf, thanks for your support and patience, specially during the last months.

Lonneke Driessen
January 2006
Eindhoven

Contents

Chapter 1

Introduction

This chapter introduces the reader to the subject of simulation-based optimization, i.e., optimization involving computer simulations as function evaluations. The design of products and processes has gradually shifted from a purely physical process towards a process that heavily relies on computer simulations (virtual prototyping). To optimize this virtual design process in terms of speed and final product quality, statistical methods and mathematical optimization techniques are widely used nowadays. Section 1.1 describes and illustrates the development of this research area 'simulation-based optimization'. The simulation-based optimization problem considered in this thesis is defined in Section 1.2. In this section we also classify the available optimization methods. One of these classes is the class of objective-driven methods, which will be further discussed in Section 1.3. We describe the general framework of these objective-driven methods and discuss different implementations. Section 1.4 concludes, listing the contributions of this thesis and giving an outline of the succeeding chapters.

1.1 Virtual product and process design

The word 'virtual' has settled in our common language in many different combinations. There is the 'virtual reality', people may work in a 'virtual company', and designers build 'virtual prototypes'. Product and process design has become virtual to a great extent. From now on, when we use the words design or product design, we refer to the design of products *and* processes. Virtual product design involves technologies that rely upon computer simulation to accurately predict product performance. The design process has changed drastically over the years, relying more and more on the use of computers. In Section 1.1.1 the trends in industry with respect to virtual product development are discussed. Thereafter, applications in television design are described in Section 1.1.2.

1.1.1 Trends in industry

The ever-increasing pressure on the development time of new products and processes has changed the design process drastically over the years. In the past, design merely consisted of experimentation and physical prototyping. These physical prototypes afford the design team an opportunity to assess the overall footprint of the product. It is hard to incorporate changes in finished prototypes though, and producing several slightly different prototypes at once is expensive. Hence, physical prototyping often is a time consuming process of entirely new prototypes being created for each iteration.

In the last decades, physical computer simulation models such as circuit design models and finite element analysis models are widely used in engineering design and analysis. The simulation models are often black-boxes, that is, only input and output are observed. Moreover they are often time consuming to run. The current reliability and stability of these Computer Aided Engineering tools has enabled the virtual prototyping of complex products and processes, a way-of-working adopted by many companies nowadays. This has led to improved product quality and performance, and reduction of product development time and costs. Another benefit of employing simulation models is that you can actually see cause and effect and track characteristics that cannot be physically measured. Virtual prototyping has changed the traditional product development cycle from 'design - build - test - fix' to 'design - analyze - test - build'. Making the physical test process a validation phase reduces time to market and is much less physical facility intensive.

Still designers are confronted with the problem of finding settings for a, possibly large, number of design variables that are optimal with respect to several simulated product or process characteristics. These characteristics may originate from different engineering disciplines. Since there are still many possible design variable settings and computer simulations are often time consuming, the crucial question becomes how to find the best possible setting with a minimum number of simulations. Usually in such a situation, designers use their intuition and experience. They carry out a number of simulation runs and choose the design that gives the best results. This intuitive approach can be considerably improved by using statistical methods and mathematical optimization techniques. Hence, the next step in the evolution of the design process has been the one towards applying these techniques. A new research field appeared: the field of simulation-based design optimization.

Simulation-based design optimization focuses on developing new and enhanced techniques for mathematical optimization in simulation environments. From now on we use the term black-box optimization instead of design optimization to stress our focus on design optimization involving black-box computer simulations. In the early nineties the first innovative companies in automotive and aerospace started to develop optimization

shells around their simulation models, often in close cooperation with universities. During the following years more and more companies using simulation models to a great extent expressed interest in optimization as a next step.

Optimization tool development shifted from universities building prototypes and dedicated solutions to newly created companies developing professional standard black-box optimization software and offering optimization support. Research groups at different universities continued to improve and extend the underlying algorithms and techniques. The recent substantial increase of consultancy firms and software vendors in this market demonstrates the growing customer demand and awareness. A few well-known dedicated optimization software packages are Compact (CQM), Design Explorer (Boeing), DOT (VR&D), Heeds (Red Cedar Technologies), modeFRONTIER (Esteco), LMS Virtual.Lab Optimization (LMS International), and Optistruct (Altair). Another recent trend is the tighter cooperation between simulation tool developers and optimization experts. As a reaction on increasing optimization support demand from clients, simulation tool developers either include optimization as a standard feature in their simulation toolbox or initiate partnerships with optimization experts.

Black-box optimization is a hot topic, both from a theoretical and a practical point of view. As an example we consider some real life applications in the next section.

1.1.2 Applications in the design of color picture tubes

The color picture tube displays the images on a television or computer monitor. More than a billion picture tubes have been manufactured to date. The basic principle of the color picture tube still equals that of its first design in 1949 (see Figure 1.1). Continuous improvement and refinement of the picture quality on the one hand, and styling on the other hand, has made the color picture tube the mature product it is today.

LG.Philips Displays (LPD) is one of the world's largest producers of cathode ray tubes (CRTs) for use in televisions and computer monitors. To stay ahead in this competitive market a focus on cost reduction and time-to-market is essential. At the same time market trends like flat TV pushes designers to the limits of the CRT concept. In this challenging environment virtual prototyping is indispensable and numerous simulation tools are used.

Over the years LPD developed several dedicated computer simulation models of picture tube parts. Designers use these models to simulate the physical behavior of a particular part design. Depending on the amount of detail in the model, the running time of a typical simulation ranges from a few minutes to ten hours. Tube designers are confronted with the problem of finding settings for a large number of design variables that are optimal with respect to several simulated tube characteristics. Examples of design variables are variables defining the screen geometry (lengths and glass thicknesses), and variables that

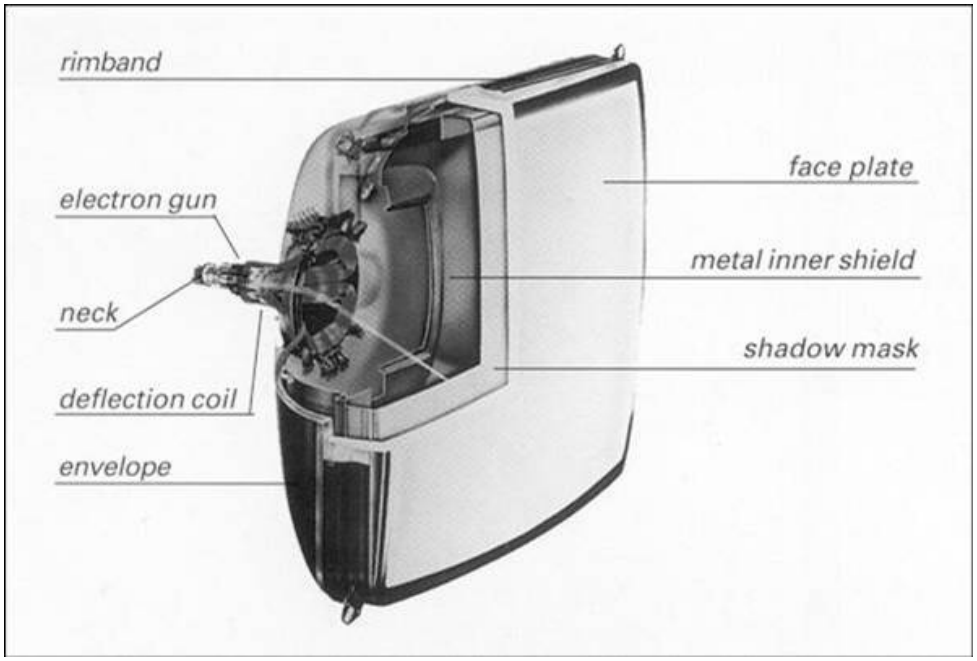

Figure 1.1: A color picture tube (source: F&GA report, Philips Components(1995)).

describe the curvature of the shadow mask. Examples of important tube characteristics are the weight of the glass, thermal and vacuum stresses in the glass, shadow mask buckling load and shadow mask displacement under local thermal load.

Optimization studies have been carried out for several picture tube parts and manufacturing processes: the screen, electron gun, and shadow mask geometry have been successfully optimized as well as the oven temperature profile for one of the manufacturing processes (Den Hertog and Stehouwer (2002a)). In the latter project for example, a reduction of 50 % was achieved for the maximal occurring stress. Several of these studies have been carried out using the optimization tool COMPACT. CQM and LPD also developed a dedicated optimization environment, named RMS-Opt, which is tightly linked with several analysis tools and is specially suited to deal with a large numbers of (possibly integer valued) design variables. This optimization program has been used for deflection coil design, and for design of electron guns. The performance of RMS-Opt has been improved by implementing several of the techniques, that are described in this thesis.

In recent years LPD has focused on supporting a more integral design process. In the architecture phase of new tubes, several part design groups work together very closely in an architecture team to translate system level specifications to part level specifications for the detailed design phase. In the architecture team, part design groups try to collaboratively meet the ever-tightening product requirements. This process is quite iteratively,

which means that successfully going through the system design phase in limited time is a difficult job. Moreover, the team has to prevent that a lot of time and energy is spent in part optimization, which is not necessarily system optimization. CQM and LPD started investigating collaborative optimization approaches that support an architecture team in managing the global tube design process systematically (see Husslage *et al.* (2003)). The approach connects part level approximation functions to obtain a system level model that gives insight in bottlenecks, trade-offs and dependencies, enables prediction of system level quality characteristics, and facilitates system level optimization and robust design.

1.2 Optimization in a simulation environment

We showed the growing interest in black-box optimization as a natural next step in virtual product design and illustrated this development for the design of picture tubes. In the next section we formally define the black-box optimization problem. Thereafter, in Section 1.2.2, we discuss the use of classical optimization methods to solve black-box optimization problems. Section 1.2.3 treats dedicated black-box optimization methods.

1.2.1 Problem formulation

The key building blocks of a black-box optimization problem are

- A set of design variables: the decision variables of the black-box optimization problem.
- A set of responses: the simulated process or product characteristics under consideration.
- A black-box simulation tool.
- An objective in terms of responses (and possibly design variables).
- A set of constraints on the design variables.
- A set of constraints on the responses.

A set of design variable values is fed to the black-box, for example a simulation tool. The simulation tool calculates the corresponding response values. The black-box process is illustrated by Figure 1.2.

Goal of the optimization problem is to minimize (maximize) a given function of the design variables and responses, subject to a set of constraints on the design variables and the responses. The functional relation between the design variables and the responses is not explicitly known. This situation often occurs when dealing with simulation models.

Figure 1.2: Schematic representation of a black-box process.

Illustration: deflection coil design

The deflection coil is an electronically controlled device that generates a strong magnetic field (see Figure 1.3). Increasing or decreasing the power supply to the coil generates a magnetic field of varying intensity. The electron beam itself is aimed by electronic manipulation of that strong magnetic field. As the coil's field interacts with the electron beam, it is deflected to make it sweep across the screen. We now discuss the key building blocks of the black-box optimization problem for deflection coil design.

Figure 1.3: A deflection coil.

Design variables: The deflection coil is produced by winding metal wire around different pins in a mould. Most important design variables are the number of windings around a coil, the x,y, and z location of the pins and the moment at which they are brought into position. The latter is an integer valued design variable, as pins are activated after a certain number of windings has been completed.

Responses: Among the most important responses of deflection coils are the errors in the size and shape of the spots created by the three distinct electron beams (red, green, and blue). When these beams do not converge in a single spot, reduced picture quality is

the result.

Black-box: A dedicated simulation tool simulates the winding process, given a setting for the design variables, and calculates the resulting image quality. Depending on the problem and application, simulation time ranges from several minutes to a quarter of an hour.

Objective: The objective is to minimize the total error. This can for example be defined as the weighted sum of quadratic errors.

Constraints on design variables: For each design variable a lower and an upper bound exists. Furthermore, there are constraints on combinations of design variables. For example, the positions of the different pins depend on each other, as well as the moments at which those pins are activated.

The black-box minimization problem can be mathematically formulated as follows

$$\min_{x,y} z = f_0(x, y, r(x, y)) \tag{1.1a}$$

$$\text{s.t. } l_i^f \leq f_i(x, y, r(x, y)) \leq u_i^f, \qquad 1 \leq i \leq m_f, \tag{1.1b}$$

$$l_j^g \leq g_j(x, y) \leq u_j^g, \qquad 1 \leq j \leq m_g, \tag{1.1c}$$

$$l^b \leq \begin{bmatrix} x \\ y \end{bmatrix} \leq u^b, \tag{1.1d}$$

$$x \in \mathbb{R}^k, y \in \mathbb{Z}^p, \tag{1.1e}$$

where $r : \mathbb{R}^{k+p} \to \mathbb{R}^v$ is the vector of responses, $f_i : \mathbb{R}^{k+p+v} \to \mathbb{R}, (0 \leq i \leq m_f)$, and $g_j : \mathbb{R}^{k+p} \to \mathbb{R}, (1 \leq j \leq m_g)$. The design variables x are real-valued, and the design variables y are integer-valued. Let $n = k + p$ be the total number of design variables. The functions $f_i (0 \leq i \leq m_f)$ and $g_j (1 \leq j \leq m_g)$ are explicitly known functions. Note that this does not imply that Problem (1.1) is explicitly known, since it still contains the unknown function r coming from the black-box tool. Moreover, it is not required that the lower bounds, l_i^f and l_j^g, and the upper bounds, u_i^f and u_j^g, in (1.1b) and (1.1c) are finite. The bounds in (1.1d) however, are assumed to be finite. The design space is defined as the area constrained by (1.1c), (1.1d), and (1.1e), which are the constraints on design variables only. Hence, the design space is known a priori. The realization of response values can be subject to certain perturbations.

Problem (1.1) differs from a standard optimization problem due to the not explicitly known functions r, which may be present both in the objective (1.1a) and in the constraints (1.1b). This means that each time we wish to evaluate a design point (x, y), we need to call the black-box tool to obtain the corresponding values for $r(x, y)$. Some black-box tools need up to a few seconds to carry out one simulation, whereas others need

several hours or even days for a single simulation.

Black-box tools also differ in the amount of noise present in the simulation outcomes. This can be stochastic as well as numerical noise (for example introduced by mesh changes in Finite Element simulation models). Classical optimization methods heavily rely on gradient estimation schemes.

Characteristics that make the class of black-box optimization problems quite complex are:

- Time consuming function evaluations.
- Absence of gradient information.
- Presence of integer design variables.
- Nonlinear objective function and nonlinear constraints.
- Existence of local minima.
- Presence of stochastic or numerical noise.

Classical optimization methods become impractical when the simulation time is considerably. They require too many time consuming function evaluations in the process of finding an optimum. Hence, depending on simulation time either classical optimization methods or special methods for black-box optimization should be used. In the next section we discuss the classical optimization methods and in the section thereafter the special methods for black-box optimization.

1.2.2 Classical optimization methods

In the former section we defined the black-box optimization problem and discussed its main characteristics. The black-box optimization problem (1.1) is a Nonlinear Programming (NLP) problem, or Mixed Integer Nonlinear Programming (MINLP) problem in case of the presence of integer design variables. Well-known standard optimization methods exist to solve the NLP problem: the generalized reduced gradient method (GRG), and sequential quadratic programming method (SQP)(see, e.g., Gill *et al.* (1981)). For MINLP problems recently methods based on branch & bound (Leyffer (2001b)) and outer approximation (Leyffer (2001a)) have been developed. Gradients play an important role in all mentioned methods.

Another group of classical optimization methods are the zero-order or derivative-free methods, like pattern search by Hooke and Jeeves, the conjugate gradient method by Powell, and the simplex method by Nelder and Mead, which do not use any gradient

information at all (see Gill *et al.* (1981) for an overview). Pattern search has been extended and used as a special method for black-box optimization by Booker *et al.* (1999) and many others.

The methods mentioned above require a considerable amount of function evaluations (and hence black-box simulations for a black-box optimization problem) during the process of locating an optimum. Therefore, classical optimization methods should only be applied to black-box optimization problems for which simulations can be carried out quickly (say in less than seconds).

In most NLP and MINLP codes, first-order or even second-order derivatives are used. Sometimes these derivatives can be calculated symbolically. For this reason, in recent years automatic differentiation has been developed (see, e.g., Griewank (1989) and Dixon (1994)). Although this is becoming more and more popular, in the case of black-box simulations this cannot be applied. In almost every NLP code, finite-difference schemes are implemented and their ability to produce good gradient estimations is important for the success of the NLP method. This holds in particular for black-box problems, for which function evaluations cannot be carried out quickly and numerical or stochastic noise may be present. Therefore, part of the research presented in this thesis is concerned with new gradient estimation schemes.

When black-box simulations are time consuming, classical optimization methods cannot be used anymore, as they require too many simulations. In this case special methods for black-box optimization should be applied. These methods are described in the next sections.

1.2.3 Special methods for black-box optimization

When black-box simulations become time consuming special methods for black-box simulations should be used. A broad range of simulation-based optimization methods exist. In this section we classify the most important ones.

We distinguish between *objective-driven* and *model-driven* methods. The focus of objective-driven methods is on finding an optimal objective value as quickly as possible. Hence, in objective-driven methods selection of simulation candidates is mainly guided by expected objective improvement. Model-driven methods on the other hand, focus on creating reliable approximation functions for the responses, and selection of simulation candidates is guided by their expected quality improvement of the approximation function. Note, that it is very well possible that an objective-driven method creates approximation functions for the responses as a part of the solution process. However, these approximation functions serve in the first place to find the optimum.

Within the class of objective-driven methods based on approximation functions we

make a further distinction between methods based on *local* approximations and methods based on *global* approximations. The focus of this thesis is on the class of objective-driven methods. Both classes are described in more detail in the next two sections.

1.2.4 Model-driven methods

Model-driven methods aim at constructing explicit global approximations for the functional relationships between the design variables and the resulting responses, which are valid throughout the whole design space. Once reliable approximation functions for the responses exist, the original, only implicitly known, response functions r can be replaced by these approximation functions, $\hat{r}$. The now explicit optimization problem can be solved using the classical optimization methods mentioned in Section 1.2.2. Hence, Problem 1.1 is replaced by the following problem:

$$\min_{x,y} z = f_0(x, y, \hat{r}(x, y)) \tag{1.2a}$$

$$\text{s.t. } l_i^f \leq f_i(x, y, \hat{r}(x, y)) \leq u_i^f, \qquad 1 \leq i \leq m_f, \tag{1.2b}$$

$$l_j^g \leq g_j(x, y) \leq u_j^g, \qquad 1 \leq j \leq m_g, \tag{1.2c}$$

$$l^b \leq \begin{bmatrix} x \\ y \end{bmatrix} \leq u^b, \tag{1.2d}$$

$$x \in \mathbb{R}^k, y \in \mathbb{Z}^p. \tag{1.2e}$$

Figure 1.4: Visual representation of a model-driven method. The second picture, which is the first picture seen from above, shows the DoE (a maximin LHD in this case).

The approximate response functions $\hat{r}$ can also be used to predict response values in certain design points, or to gain insight in relations between design variables and responses, for example through 2D and 3D visualization. Furthermore, robustness studies can be carried out without the need for many time consuming simulations.

Another advantage of having reliable approximation functions available, is that the objective function f_0 and constraints $f_i, (1 \leq i \leq m_f)$ can be altered without the need

for additional simulations. As long as the design space does not change, the global approximation functions remain valid. Hence, a new explicit optimization problem can be solved, without the need for extra time consuming simulations.

Lastly, we mention sensitivity analysis: it has often proven to be very useful to investigate the impact of the value of a particular lower or upper bound on the value of the objective function of the optimal design. Such bound sensitivity analysis involves re-calculation of the optimal design point for a number of successive bound values, which can be easily done using the explicit approximate response functions. Note that these calculations are only valid within the bounds of the original design space.

The disadvantage of global approximation functions, however, is the high number of simulations required to obtain good global approximations. This required number of simulations grows exponentially with the number of design variables. For design problems containing more than 20-25 design variables, the use of global approximation functions therefore becomes impractical.

Model-driven methods differ from each other mainly in two areas, namely the selection of the initial simulation scheme or Design of Experiments (DoE) (and possibly later additional simulation points) and the chosen approximation function type (see Figure 1.4). Specific DoE's for computer simulation exist, e.g., the so called space-filling designs. An example of those are the maximin Latin Hypercube Designs (LHDs) proposed in Morris and Mitchell (1995). Statistical DoE's, like D-optimal schemes and full or fractional factorial schemes, are also used. In case it turns out that the fit of the obtained approximation functions is not satisfactory yet, additional simulations can be carried out, one by one or as an additional DoE. In Van Beers and Kleijnen (2004) a novel application-driven sequential method to select an experimental design is proposed. We refer to Koehler and Owen (1996) and Santner *et al.* (2003) for a treatment of computer experiments. Specific DoE's for computer simulation often use the geometry of the design points as leading measure. A set of points has a good geometry when the points are located throughout the design space in such a way that they offer enough information in all dimensions of the design space. Note that the concept geometry relates to the design space only. Hence, it does not incorporate measures with respect to the goodness of fit of the approximation function in any way.

Within the class of model-driven derivative-free methods we find among others the following types of global approximation functions:

- **Regression models:** The approximation functions are obtained by linear or non-linear regression. Most often they are first- or second-order polynomials. Myers and Montgomery (1995) discuss this response surface methodology. See ,e.g., Schoofs *et al.* (1994) for an application in the design of bells, and Den Hertog and Stehouwer

(2002a) for an application in color picture tubes.

- **Neural networks:** A neural network is a system composed of many simple processing units, which are wired together in a complex communication network. A neural network is able to predict an output pattern when it recognizes a given input pattern. Once trained, the neural network is able to recognize similarities when presented with a new input pattern, resulting in a predicted output pattern. Neural networks can therefore be used as approximation functions. See Bishop (1995) for a treatment of neural networks. Laguna and Martí (2002) use neural networks as approximation functions in the context of simulation optimization.

- **Symbolic regression models:** In symbolic regression no specific model is assumed yet and genetic programming is used to find the best fitting symbolic description of the approximation functions (Koza (1994)). Ashour *et al.* (2003) use symbolic regression models in the design of reinforced concrete deep beams.

- **Rational functions:** A rational function is formed when a polynomial is divided by another polynomial. Rational functions have been applied as approximation functions in Cuyt *et al.* (2004) and Dhaene (2002).

- **Radial Basis functions:** Radial Basis functions are a specific type of neural networks, often used for interpolation and approximation. Powell (2001) reviews them. Shen *et al.* (2002) use Radial Basis functions to develop a nonlinear temperature model of molten carbonate fuel cell systems.

- **Kriging models:** In a Kriging model the response function $r_i(x)$ is treated as a realization of a stochastic process, just like in linear regression. A difference between a Kriging model and a regression model is that the Kriging model uses spatial correlation: the correlation between two data points is larger as they lie closer together. Another difference is that Kriging functions are interpolating. In fact the Kriging function is the Best Linear Unbiased Predictor. Sacks *et al.* (1989) introduce Kriging models for computer-aided design, Kleijnen and Van Beers (2004) give a survey of Kriging models in simulation, and, e.g., Den Hertog and Stehouwer (2002a) apply them in the design of color picture tubes.

1.2.5 Objective-driven methods

The primary aim of objective-driven methods is to find the global optimum as quickly as possible. To this end often local or global approximation functions are built. We therefore classify the objective-driven methods based on whether they use approximation and if so,

on what level. Unlike for model-driven methods, for objective-driven methods additional time consuming simulations need to be carried out as soon as the optimization objective or constraints are altered. The optimum is probably relocated, and as the objective-driven local or global approximation functions most likely do not show a good fit throughout the whole design space, additional simulations are carried out in search for the new optimum. On the other hand, objective-driven methods can be applied to problems with a large number of design variables, and they find good points early in the optimization process. When the underlying real model is very nonlinear, objective-driven methods can still be applied in combination with a multistart approach, whereas model-driven methods might not be able to capture the underlying model well enough.

Methods using local approximation functions

These methods rely on low-order interpolation or linear regression models in a trust region framework. The trust region is the area in which the local approximation functions are expected to predict well. To construct the required low-order approximations only a small initial DoE is needed, compared to the DoE for global model-driven methods. Therefore, this approach can be applied to optimization problems with more than 20-25 design variables as well. In the illustrative example of deflection coil design in Section 1.2.1 the design problem consists of 70-90 design variables. The local approximation functions are constructed using (weighted) regression techniques. The approximated objective function is then optimized within the trust region to find the best feasible objective improving point. In each iteration new local approximation functions are built, and a new point is evaluated and/or the trust region is decreased/increased. Convergence is guided by the size of this trust region. The process is illustrated in Figure 1.5. In each iteration the following explicit problem is solved, replacing Problem 1.1:

$$\min_{x,y} z = f_0(x, y, \hat{r}(x, y)) \tag{1.3a}$$

$$\text{s.t. } l_i^f \leq f_i(x, y, \hat{r}(x, y)) \leq u_i^f, \qquad 1 \leq i \leq m_f, \tag{1.3b}$$

$$l_j^g \leq g_j(x, y) \leq u_j^g, \qquad 1 \leq j \leq m_g, \tag{1.3c}$$

$$l^b \leq \begin{bmatrix} x \\ y \end{bmatrix} \leq u^b, \tag{1.3d}$$

$$(x, y) \in TR, \tag{1.3e}$$

$$x \in \mathbb{R}^k, y \in \mathbb{Z}^p, \tag{1.3f}$$

where TR denotes the trust region. The difference with Problem (1.2) lies in the additional constraints (1.3e), the trust region constraints. They could be defined as box constraints, or as spherical or ellipsoidal constraints for example.

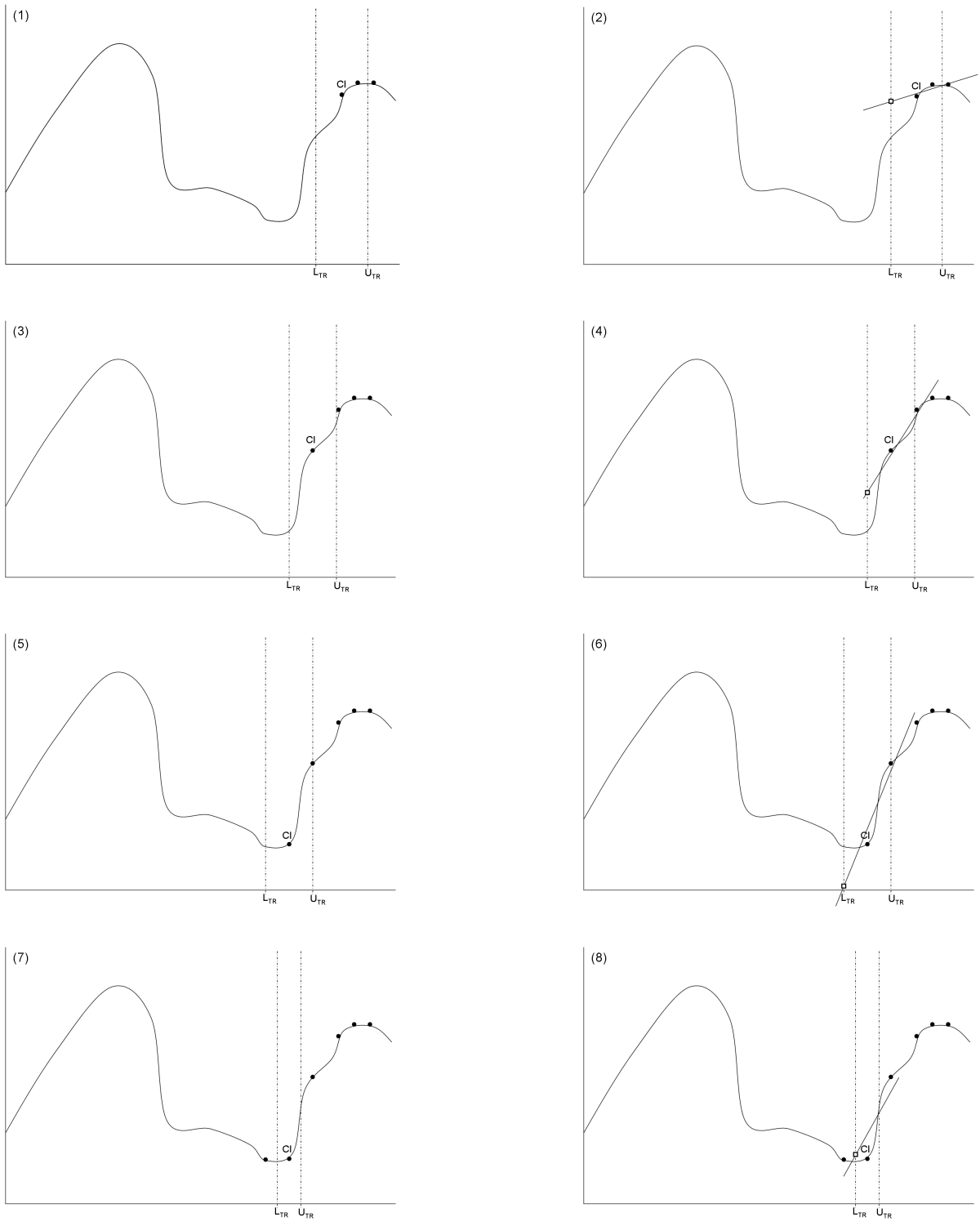

Figure 1.5: Visual representation of an objective-driven method using local linear approximations for a 1-dimensional design space. Picture (1) shows the DoE, the current iterate (the center of the trust region), denoted as CI, and the trust region bounds, denoted as L_{TR} and U_{TR}. Picture (2) shows the local linear approximation function and the predicted optimum. This point is simulated, resulting in the situation in Picture (4), and so on. In Picture (7) the trust region is reduced, as the former step did not result in a better point.

A few references for objective-driven methods using local approximation functions are Powell (1994a), Conn and Toint (1996), Toropov (1999), Marazzi and Nocedal (2002), and Brekelmans *et al.* (2005). In Section 1.3.2 we will discuss them in more detail.

Methods using global approximation functions

Just like model-driven methods, these methods often rely on complex, sophisticated global approximation functions. The initial DoE is usually smaller, though. The idea is to add design points iteratively and construct new approximations every time a simulation has been carried out (and hence, new information becomes available). Some of these simulations are aimed at objective improvement, while others are aimed at improving the accuracy of the approximation functions to avoid getting trapped in local optima. The mathematical program solved in each iteration is Problem (1.2). Figure 1.6 gives a visual representation.

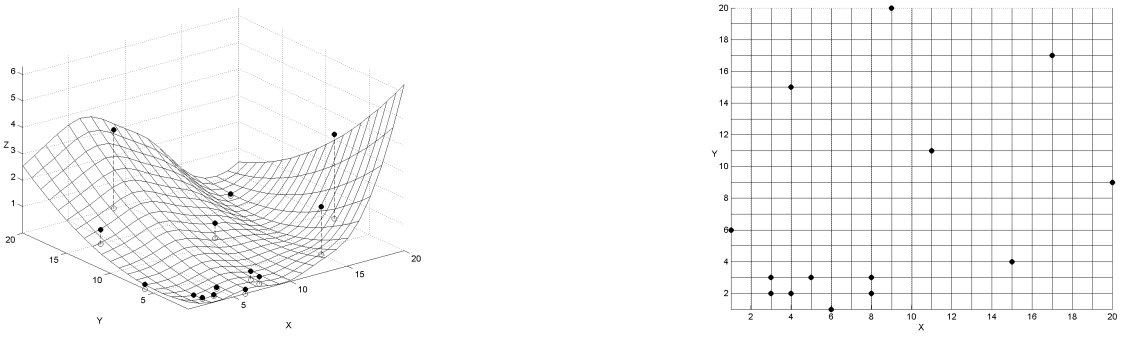

Figure 1.6: Visual representation of an objective-driven method using global approximations. Though the DoE is nicely spread over the design space, in later iterations typically simulations are concentrated in interesting areas.

References for objective-driven methods using global approximation functions are Jones *et al.* (1998), discussing the use of Kriging models, and Gutmann (2001) and Björkman and Holmström (2000), using Radial Basis functions. Jones (2001b) gives a good overview of objective-driven methods using global approximation functions. These methods are discussed in more detail in Section 1.3.2.

Some other methods. A special class of methods are the pattern search methods described by Dennis and Torczon (1994) and Booker *et al.* (1999). This extension of the classical pattern search (Hooke and Jeeves (1961)) is a framework consisting of a search step and a poll step. The poll step ensures convergence and in the search step any search strategy can be incorporated, including any of the objective-driven methods described above.

Bandler *et al.* (1995) present the Space Mapping method. Unlike the other methods, the Space Mapping method constructs a parameter mapping between the black-box simulation model and the approximation function. Later, Bakr *et al.* (1998) place the Space Mapping method in a trust region framework, using local approximation functions instead of global ones.

Genetic Algorithms (Holland (1975)), Simulated Annealing (Aarts *et al.* (1997)), and Tabu Search (Glover and Laguna (1997)) form a class of probabilistic local search metaheuristics to solve global optimization problems. These methods do not require any derivative information and can be used to solve simulation-based optimization problems. These methods have in common that they require a lot of simulations to come to a good solution. Therefore, we do not consider them as appropriate methods to use in case of time consuming black-box simulations. The DIRECT algorithm (Jones *et al.* (1993)) is another well known method using no approximation functions for continuous optimization problems. A later extension handles nonlinear and integer constraints (see Jones (2001a)).

This thesis considers objective-driven methods and building blocks therein. In the next section we describe a general framework for objective-driven methods. Then we discuss for each building block of the framework how it has been implemented in different methods, both from literature and own research.

1.3 Survey of objective-driven methods

Almost all objective-driven optimization methods roughly follow the same sequence of steps. In Section 1.3.1 we describe this framework. In Section 1.3.2 we discuss a set of well-known objective-driven methods as well as our own contributions, within the context of this framework.

1.3.1 General framework

Figure 1.7 shows a flowchart of objective-driven optimization methods. Unlike Jacobs (2004) we incorporate objective-driven methods based on global approximations in our framework. We discuss each of the steps below.

Step 1 *Problem formulation and initialization*

First a complete problem formulation should be given: what are the design variables, which responses are important, what constraints should be imposed and what is the objective function, which black-box tools are needed? The importance of choosing suitable design variables and responses should not be underestimated. Often it is not obvious which design variables must be taken into account. The following

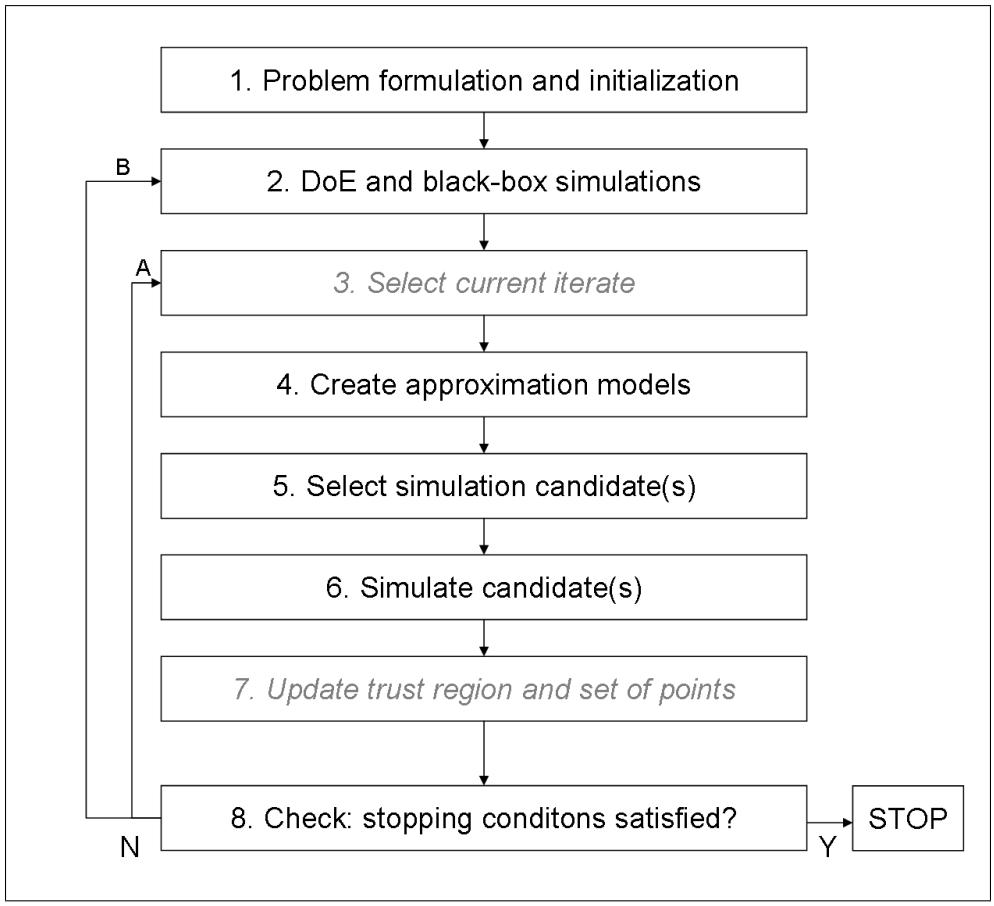

Figure 1.7: Flowchart of objective-driven optimization methods.

aspects should be taken into consideration (as described in our paper Stinstra *et al.* (2001)):

(a) A too complex parameterization should be avoided, but at the same time it must be complex enough to represent the real-life design in a satisfying way. Suitable transformations of design variables or responses can prevent the need for highly nonlinear approximation functions.

(b) A redundant parameterization should also be prevented. For example, symmetry may cause multiple different runs to specify the same design. In such a parameterization, usually a too large design space is modelled.

(c) Another pitfall that should be avoided is to model compound responses. Compound responses are known functions of the response parameters, that arise in the objective f_0 or the constraints f_i of the optimization problem. For example, when minimizing the maximum over a number of responses, one might choose to define the compound response $r(x, y) = \max_i r_i(x, y)$ and generate an approximation function for this compound response. In our opinion this is the wrong approach, as it enforces the approximation model to re-discover the function structure of the compound response. It is better to generate an approximation function for each response r_i separately and define $f_0 = \max_i r_i(x, y)$.

In this step, also some initialization takes place, for example, the choice of the initial trust region size, and the design point to center the trust region at, also named the current iterate.

Step 2 *DoE and black-box simulations*

This step consists of generating a suitable DoE and carrying out the black-box simulations accordingly. In general, the DoE is larger for methods based on global approximations than for methods based on local approximations. For the latter class the DoE is located within (the close neighborhood of) the trust region. In the presence of integer design variables, the DoE should of course account for the integer nature of these variables.

Step 3 *Select current iterate*

This step only applies to methods based on local approximations. The area in which the approximation functions are assumed to be valid, often referred to as the trust region, differs in size and location between iterations. The trust region is centered around the current iterate, and hence, in each iteration the decision about which of the simulated points becomes the current iterate needs to be made. An obvious choice would be to take the design point with the best objective value so far as

current iterate. As for methods based on global approximations no trust region applies, they do not need a current iterate neither.

Step 4 *Create approximation functions*
Next, the approximation functions are constructed. As discussed before, there are many possible choices of approximation functions, some based on regression techniques, some based on interpolation. For methods based on local approximations often not all points that have been simulated are used to construct the approximation functions: either because they are located too far away from the trust region or because the approximation functions are interpolating polynomials that require a fixed number of simulation points. It is also possible to apply a weight scheme when constructing the approximations, to give more importance to certain design points than to others. The result of this step should be a set of approximation functions for the responses, with accurate prediction capabilities for the part of the design space in which they are valid.

Step 5 *Select simulation candidate(s)*
By replacing the black-box relation between design variables and responses by the explicit approximation functions, classical optimization subproblems arise. They can be aimed at objective improvement and/or approximation function improvement. Solving one or more of such optimization problems results in a set of interesting design points, candidates for simulation. Some methods select one candidate, others allow for selection of several candidates.

Approximation function improvement can be pursued in different ways. Existing methods can be divided in two categories, depending on whether they focus on the design space, or on the realized fit of the approximation functions. The first category aims at geometry improvement, hence improvement of the dispersion of the set of design points throughout the design space. The second one uses measures of goodness of fit of the current approximation function (for example prediction error) to improve the fit in the areas where it needs improvement most.

Step 6 *Simulate candidate(s)*
The selected candidate(s) are simulated.

Step 7 *Update trust region and set of points*
Also this step only arises in methods using local approximation functions. Based on the simulation results, the size of the trust region is reconsidered and possibly increased or decreased. The set of points on which the approximation functions

should be based in the next iteration is also reconsidered. For polynomial interpolating models for example, this set consists of a fixed number of design points and every time newly simulated design points are added to the set, other design points have to leave the set to keep it at the same size.

Step 8 *Check: stopping conditions satisfied?*
In this step the stopping conditions are evaluated. If they are met, the algorithm terminates and the best design point found so far is returned. If the stopping conditions are not met, the next iteration is entered. Methods based on global approximations start the next iteration at Step 4 (via arrow A to Step 3, which they skip). Some methods based on local approximations return to Step 2 (arrow B) and create a totally new DoE to start from in each iteration, while others return to Step 3 (arrow A). Possible stopping conditions for example relate to the number of simulations carried out or the size of the trust region.

This is a general, high-level description of objective-driven methods. In the next section we zoom in to each of these steps and discuss different implementations with references to literature.

1.3.2 Discussion of existing and new implementations

In the literature we encounter different implementations for each of the steps listed in Figure 1.7. This section lists the most important ones. In addition, we place our own work within the framework.

Step 1 *Problem formulation and initialization*
This first step is in essence the same for all methods. We already stressed the importance of a good problem formulation. Furthermore, for each method different parameters and/or sets need to be initialized.

Step 2 *DoE and black-box simulations*
The initial DoE is important for the quality of the approximation functions. We first discuss some DoE approaches used for objective-driven methods based on local approximations.

Several of these methods create a complete new DoE in each iteration. These methods are known as multipoint approximation methods (Toropov *et al.* (1993), Abspoel *et al.* (2001), Craig and Stander (2002)). Starting from a new DoE in each iteration ensures a good geometry of the points on which the local approximation functions are based. Therefore, these methods only select objective improving candidates in Step 5. Toropov *et al.* (1993) uses a simple DoE of n new points, that are obtained

by a perturbation of the current iterate: each design variable is perturbed by a fraction of the corresponding size of the trust region. Later, Toropov (1999) states that such scheme works only well in conjunction with intrinsically linear approximations (i.e., nonlinear approximation functions that can be unambiguously transformed to functions that are linear in the newly defined variables.). As disadvantages he names the fact that the scheme does not take into account design points already present in the area and the fact that the scheme is inflexible with respect to the number of added points. He introduces a new scheme which distributes points as homogeneously as possible through the trust region by introducing a certain cost function. This cost function tries to both maximize the distance between points and create a homogeneous distribution along the coordinate axes. Abspoel *et al.* (2001) create a D-optimal design as initial set. Angün (2004) extended the RSM approach for multiple responses in a stochastic simulation environment.

Conn and Toint (1996) randomly generate points in a sphere centered at the initial current iterate until the set of points they need for interpolation is poised (i.e., an interpolant exists). Then they use their geometry improving sub-method to improve the geometry of this initial set. Powell (1994a) forms a non-degenerate simplex to start from.

Marazzi and Nocedal (2002) define the initial set by

$$x^i = x_c \pm \Delta_c e_i, \qquad i = 1, ..., n,$$

where Δ_c denotes the initial trust region radius, and e_i the i^{th} canonical vector, and the sign $\pm$ is chosen randomly.

Powell (2002) chooses n points similarly to Marazzi and Nocedal (2002) (only the $\pm$ is replaced by a $+$), another n points either $x_c + 2\Delta_c e_i$ (when the objective value in $x_c + \Delta_c e_i$ is lower than in x_c) or to the negative side (when the objective value in $x_c + \Delta_c e_i$ is higher than in x_c). The remaining necessary interpolation points are perturbed in two dimensions instead of one. These choices make it easy to derive the parameters of the first quadratic approximation functions.

As for the objective-driven methods based on global approximation functions, Koehler and Owen (1996) give a good overview of DoE's for computer experimentation and discuss various criteria that could be used to create such a DoE. Björkman and Holmström (2000) initiate their method with the 2^n corners of their box-constrained design space. Jones *et al.* (1998) initiate their method with a space-filling DoE, a Latin Hypercube Design that has the property that all the one- and two-dimensional projections are nearly uniformly covered.

Brekelmans *et al.* (2005) (see Chapter 4) propose to use a D-optimal design as DoE for continuous black-box optimization problems. Driessen *et al.* (2005) (see Chapter 6) adapt a space-filling LHD for continuous black-box optimization problems to being usable also for integer black-box optimization problems.

Step 3 *Select current iterate*

The most common choice for the current iterate is the best design point found so far and the trust region is usually centered at this current iterate (Conn and Toint (1996), Marazzi and Nocedal (2002), Powell (2002)). The multipoint strategy by Toropov *et al.* (1993) takes the best point found so far as corner point of the trust region in iteration $i + 1$, if this best point was on the border of the trust region in iteration i.

Brekelmans *et al.* (2005) propose to use the filter method (Fletcher and Leyffer (2002)) to judge design points by their objective value and constraint violations simultaneously. In this way also promising infeasible design points are allowed to become current iterate. If an infeasible design point is encountered it may very well be closer to the optimum than the old current iterate, and thus it should provide a promising step on the path towards the optimum. The basic principle of the use of a filter for the selection of the current iterate is that the last evaluated design point becomes the current iterate if it is accepted in the filter. If the design point is dominated by any of the previously evaluated designs, then the current iterate remains unchanged.

The location of the ellipsoidal trust region proposed in Driessen *et al.* (2006) does not depend on a current iterate, but on the locations of all points that are used to create the approximating models. This ellipsoidal trust region eliminates the need for a current iterate.

Step 4 *Create approximation functions*

In Section 1.2.3 we described several choices for approximation functions. In the literature on objective-driven methods we encounter for methods based on local approximations, linear interpolation models (Powell (1994a), Marazzi and Nocedal (2002)), quadratic interpolation models (Conn *et al.* (1997), Powell (2002), Marazzi and Nocedal (2002)), and (weighted) linear regression models (Toropov (1999), Abspoel *et al.* (2001), Brekelmans *et al.* (2005)). Toropov *et al.* (1993) also uses intrinsically linear functions (for example, multiplicative functions) as approximation functions. Powell (2003) develops a method to construct quadratic interpolation functions based on less than $\frac{1}{2}(n + 1)(n + 2)$ points. The arising freedom in the quadratic model is taken up by minimizing the Frobenius norm of the second deriva-

tive matrix of the change to the model.

Methods based on global approximations mainly use Kriging models (Jones *et al.* (1998), and Driessen *et al.* (2005)) and Radial Basis functions (Björkman and Holmström (2000), Gutmann (2001), and Regis and Shoemaker (2005)). Booker *et al.* (1999) incorporated Kriging approximations into the search step of the pattern search algorithm. Simpler approximation functions, like quadratic surfaces, are also used, but they are unable to capture nonlinear behavior well, and adding additional points will not necessarily lead to a more accurate approximation. As in some situations gradient information is available at little additional cost, several methods include gradient information when constructing the approximation functions (see for example Van Keulen and Vervenne (2002)).

Step 5 *Select simulation candidate(s)*

In most methods two different criteria play a role when selecting simulation candidates: objective improvement and approximation function improvement. We first consider the methods based on local approximations.

The methods of Powell (1994a) and Conn *et al.* (1997) first find the predicted local optimum by solving the approximated optimization problem within the trust region. If the predicted objective improvement is too small or replacing one of the sample points by this new point violates the geometric conditions, an approximation function improvement step is entered and the resulting design point is proposed for simulation instead of the objective improving point. The measure they use in the approximation function improvement step, is strongly related to the determinant of the design matrix.

Instead of solving a standard trust region subproblem and taking special action if the new point does not enjoy favorable geometric properties, Marazzi and Nocedal (2002) explicitly impose a geometric condition, the wedge constraint, in the step computation procedure, thereby guaranteeing that the new set of points defines an adequate model.

Toropov *et al.* (1993) and other multipoint methods incorporate geometry preservation by starting each iteration with generating a new DoE. This can be a very costly way of geometry preservation.

Brekelmans *et al.* (2005) use the filter principle once more to generate a set of interesting design points from objective perspective. This set contains feasible as well as infeasible points. Their set of geometry improving points is built up using the D-optimality criterion described in Driessen *et al.* (2006). From the combined set of geometry and objective improving points they select the point that, given a

required minimal performance with respect to the geometry, has the best possible expected filter improvement.

Conn *et al.* (2004) and Conn *et al.* (2005) discuss the dispersion of design points for polynomial interpolation, polynomial regression, and underdetermined interpolation models. They base the approximation function improvement measure on an explicit expression for the error between the approximation and the true function.

Methods based on global approximations also alternate between objective improvement and approximation function improvement steps, or local and global search, respectively. Gutmann (2001) and Björkman and Holmström (2000) define a utility function reflecting the error between Radial Basis function approximations and the real black-box model, as well as what they call a measure of 'bumpiness' of the Radial Basis functions. The next point, where the original objective function should be evaluated, is the point where the utility function is maximal. Jones *et al.* (1998) define the 'expected improvement' utility function as a figure of merit to balance local and global search. In each iteration they maximize expected improvement using a branch- and bound algorithm. Regis and Shoemaker (2005) also use Radial Basis functions as approximation functions. They include an extra constraint in the optimization problem. This constraint forces the new point to maintain a certain minimal distance to already simulated points. In subsequent iterations the minimal required distance is varied, resulting in iterations focussed on local or global search.

Step 6 *Simulate candidate(s)*

Often the set of candidates contains only one candidate. As this step is the most time consuming step of an iteration, and as simulations can be very well carried out in parallel, this step can be sped up considerably by means of parallelization in case multiple simulations have to be carried out. Vanden Berghen and Bersini (2005) propose a parallel constrained extension of Powell's UOBYQA algorithm. This extension uses parallel processes to increase the quality of the approximation functions by sampling well-chosen additional points. Sóbester *et al.* (2004) use Radial Basis functions as global approximation functions and simulate several promising points (with respect to *expected* improvement) in parallel.

Step 7 *Update trust region and set of points*

Once the proposed candidate has been simulated, the predicted response values can be compared with the real response values and depending on the outcome of this comparison, choices about the size of the trust region and set of points to base the approximation functions on are made. The multipoint methods basically use the points arising from the latest DoE to build approximation functions with. The trust

region is decreased if the current iteration did not result in an improved objective function.

Marazzi and Nocedal (2002) increase the trust region when the achieved objective improvement is a certain factor larger than the predicted objective improvement, otherwise they reduce the trust region. Furthermore they add the newly simulated point to the set if it improves the objective function or if it is not furthest away from the current iterate. This newly simulated point replaces the point furthest away from the current iterate. This way they promote the conservation of trial points in the vicinity of the current iterate.

Conn and Toint (1996) and Powell (2002) also look at the quotient of the predicted and realized objective improvement. If the realization is much better than expected, they compute a long step in which they temporarily increase the trust region and simulate the predicted optimum in this larger trust region. Their geometry measure is based on the determinant of the design matrix. If the newly simulated point improves the objective function or the geometry enough, they remove a point from the set of interpolation points and replace it by the newly simulated point. If this geometry improvement step fails, the trust region is reduced.

Brekelmans *et al.* (2005) choose to use weighted regression as model fitting technique. The weights of the different points depend on their distance to the current iterate. This is another way to ensure that the approximations are based on the points closest to the current iterate.

Step 8 *Check: stopping conditions satisfied?*
The simplest stopping condition would be to terminate the algorithm as soon as a certain maximum number of black-box simulations has been carried out. This condition, however, does not take into account the state of the algorithm. Conn and Toint (1996) use two stopping conditions; if the trust region size is small enough or the objective value is close enough to a given lower bound, their algorithm terminates. Powell (1994a) and Marazzi and Nocedal (2002) also uses the size of the trust region as stopping condition. Toropov *et al.* (1993) terminate when the approximations are accurate enough and the trust region has reached a predefined size and the best point found so far is not on the boundary of this trust region. Abspoel *et al.* (2001), who propose a method based on local approximations for stochastic systems with integer variables, terminate when either the trust region radius becomes smaller than 1 (because the integer variables cannot be altered any longer in that case), or the objective function does not significantly in- or decrease any longer. Jones *et al.* (1998) use their concept of expected improvement also in the stopping conditions:

as soon as the expected improvement is smaller than 1% their method terminates.

For each of the main steps of objective-driven simulation-based optimization methods we discussed different implementations. In the next section we point out the contributions of this thesis to the field of simulation-based optimization.

1.4 Contributions and outline of the thesis

The main contributions of this thesis are in two areas. The first is the area of gradient estimation for black-box optimization problems for which a single simulation can be carried out quickly, say within seconds. For these optimization problems classical optimization methods can be used. Classical methods heavily rely on gradient estimations. The ability of the finite-difference scheme to produce good gradient estimations is important for the success of the classical methods. This holds in particular for black-box problems, for which function evaluations cannot be carried out quickly and numerical or stochastic noise may be present. We therefore propose and analyze different (new) gradient estimation schemes for use in classical optimization methods.

The second is the area of objective-driven methods for time consuming black-box optimization problems. We propose a new objective-driven method based on local approximation functions and a new geometry and trust region concept. We discuss simulation-based optimization methods for integer valued problems, and apply an objective-driven method to the design of heat sinks.

The remaining chapters of this thesis consist of research papers on these subjects. As a result, notation introduced and used in one chapter may not be in line with notation introduced and used in another chapter. Section 1.4.3 gives an overview of these research papers. In Section 1.4.1 and Section 1.4.2 we discuss the main contributions of the thesis and refer to the corresponding chapters.

1.4.1 New gradient estimation schemes

In Chapter 2 we analyze different schemes for obtaining gradient estimates when the underlying functions are noisy. As an error criterion we take the mean-squared error. For three finite-difference schemes and two DoE schemes, we analyze both the deterministic errors, also known as bias, and the stochastic errors. We also derive optimal stepsizes for each scheme, resulting in a minimal total error. Central finite differencing and replicated central finite differencing schemes have the nice property that this stepsize also minimizes the variance of the error. Based on these results and on the successful application of Plackett-Burman schemes in the field of assets and liability management, we conclude

that it is worthwhile to use DoE schemes to obtain good gradient estimates for noisy functions.

In Chapter 3 we analyze the use of Lagrange interpolation polynomials for obtaining gradient estimations in the presence of stochastic or numerical noise. We analyze the mean-squared error using (N times replicated) Lagrange interpolation polynomials. We show that the mean-squared error is of order $N^{-1+\frac{1}{2d}}$ if we replicate the Lagrange estimation procedure N times and use $2d$ evaluations in each replicate. As a result the order of the mean-squared error converges to N^{-1} if the number of evaluation points increases to infinity. We also provide an optimal division between the number of grid points and replicates in case the number of evaluations is fixed. Further, it is shown that the estimation of the derivative is more robust when the number of evaluation points is increased. Test results show that schemes based on Lagrange polynomials outperform central finite-differencing for low noise levels and boil down to central finite-differencing for high noise levels.

1.4.2 Objective-driven methods

Chapter 4 presents a new objective-driven method based on local approximation functions for black-box optimization problems with time consuming function evaluations. This method has been implemented in the toolbox SEQUEM. Some of the key elements of the algorithm are inspired by the filter method (Fletcher and Leyffer (2002)), which enables the optimization of a constrained problem without the use of a penalty function. Furthermore, the filter provides the designer with a set of possible attractive design points to choose from instead of just a single optimal design. The quality of the local approximations is safeguarded by a geometry measure for the locations of the simulation points. Most steps of the algorithm are to a great extent independent of each other. Therefore, it is possible to use different implementations for these steps. Several of the in this chapter described techniques have been successfully implemented in RMS-Opt (see Section 1.1.2).

Chapter 5 analyzes two important building blocks of objective-driven methods based on local approximation functions: the trust region and the preservation of a good geometry of the design points. We propose to incorporate the D-optimality criterion, well-known in design of experiments, into these methods in two different ways. Firstly, it is used to define an ellipsoidal trust region, that adapts its center point and shape to the locations of the design points on which the approximation functions are based. Secondly, it is used to determine an optimal geometry improving point. We show the intuition behind the incorporation of the D-optimality criterion. Furthermore, we prove the independency of affine transformations for the ellipsoidal trust region. The proposed trust region and geometry improvement can both be implemented in existing objective-driven methods

based on local approximation functions.

In recent publications, optimization problems with time consuming function evaluations and integer variables have been solved using objective-driven methods based on *local* approximation functions (see for example Bremicker *et al.* (1990), Loh and Papalambros (1991b), Abspoel *et al.* (2001), and Jacobs (2004)). In Chapter 6 we advocate the use of optimization methods based on *global* approximation functions for such optimization problems. Note that this is only possible when the number of design variables is not too high and the response behavior is not extremely nonlinear. We show that methods based on local approximations may lead to the integer rounding of the optimal solution of the continuous problem, and even to worse solutions. Then we discuss a method based on global approximations. Test results show that such a method performs well, both for theoretical and practical examples, without suffering the disadvantages of methods based on local approximations.

The successful application of an objective-driven method to the design of a heat sink is described in Chapter 7. Heat sink selection is important from a business perspective, as a reduction in heat sink mass can represent a significant overall cost saving in high volume products. The black-box simulations (in this case Computational Fluid Dynamics calculations) for this case are extremely time consuming. The developed method relies on local approximations despite the presence of integer design variables. This choice was made, because due to the object collision constraints the design space consists of many disconnected areas. The responses are expected to behave very nonlinearly, in particular near the infeasible parts within the design space. Therefore we decided to use a method based on local approximations. Global search is enforced by starting several optimization sequences from different starting points. The results illustrate the efficiency of the reported methodology in finding the optimum design.

1.4.3 Overview of research papers

The remaining chapters of this thesis consist of the following papers:

Ch. 2 Gradient estimation schemes for noisy functions
R.C.M. Brekelmans, L.T. Driessen, H.J.M. Hamers, D. den Hertog
Journal of Optimization Theory and Applications,
(2005) 126(3), 529 – 551

Ch. 3 Gradient estimation by Lagrange interpolation schemes
R.C.M. Brekelmans, L.T. Driessen, H.J.M. Hamers, D. den Hertog
CentER Discussion Paper 2003-12

Ch. 4 Constrained optimization involving expensive function evaluations: a sequential approach
R.C.M. Brekelmans, L.T. Driessen, H.J.M. Hamers, D. den Hertog
European Journal of Operational Research,
(2005) 160(1), 121 – 138

Ch. 5 On D-optimality based trust regions for black-box optimization problems
L.T. Driessen, R.C.M. Brekelmans, H.J.M. Hamers, D. den Hertog
Journal on Structural and Multidisciplinary Optimization,
(2006) 31(1), 40 – 48

Ch. 6 Why methods for optimization problems with time consuming function evaluations and integer variables should use global approximation models
L.T. Driessen, M. Gerichhausen, R.C.M. Brekelmans, H.J.M. Hamers, D. den Hertog
CentER Discussion Paper 2006-04

Ch. 7 Simulation-based design optimization methodologies applied to CFD
J. Parry, R.B. Bornoff, H.P. Stehouwer, L.T. Driessen, E.D. Stinstra
IEEE Transactions on Components and Packaging Technologies,
(2004) 27(2), 391 – 397

Chapter 2

Gradient estimation schemes for noisy functions

Abstract: In this paper we analyze different schemes for obtaining gradient estimates when the underlying functions are noisy. Good gradient estimation is important e.g. for nonlinear programming solvers. As error criterion we take the norm of the difference between the real and estimated gradients. The total error can be split into a deterministic error and a stochastic error. For three finite-difference schemes and two design of experiments (DoE) schemes, we analyze both the deterministic errors and stochastic errors. We derive also optimal stepsizes for each scheme, such that the total error is minimized. Some of the schemes have the nice property that this stepsize minimizes also the variance of the error. Based on these results, we show that, to obtain good gradient estimates for noisy functions, it is worthwhile to use DoE schemes. We recommend to implement such schemes in NLP solvers.

2.1 Introduction

We are interested in a function $f : \mathbb{R}^n \to \mathbb{R}$ and more specifically its gradient $\nabla f(x)$. The function f is not explicitly known and we cannot observe it exactly. All observations are the result of function evaluations, which are subject to certain perturbation errors. Hence, for a fixed $x \in \mathbb{R}^n$ we observe an approximation

$$g(x) = f(x) + \varepsilon(x). \tag{2.1}$$

The error term $\varepsilon(x)$ represents a random component. We assume that the error terms in (2.1) are i.i.d. random errors with $E[\varepsilon(x)] = 0$ and $V[\varepsilon(x)] = \sigma^2$; hence, the error terms do not depend on x. Note that g can be also a computer simulation model. Even deterministic models are often noisy due to all kinds of numerical errors.

In this paper, we analyze both finite-difference schemes and design of experiments (DoE) schemes for obtaining gradient estimations. In all these schemes, the gradient is estimated by observing the function value in several points in the neighborhood of x, using finite stepsizes h. We compare the resulting errors made in the gradient estimations due to both the presence of noise and the deterministic approximation error (lack of fit). It will appear that DoE schemes are worthwhile alternatives for finite-difference schemes

in the case of noisy functions. Moreover, we will derive efficient stepsizes for the different schemes, such that the total error (sum of the deterministic and stochastic errors) is minimized. We compare these stepsizes to those which minimize the variance of the total error.

Gradients play an important role in all kinds of optimization techniques. In most nonlinear programming (NLP) codes, first-order or even second-order derivatives are used. Sometimes these derivatives can be calculated symbolically; in recent years automatic differentiation has been developed; see, e.g., Griewank (1989) and Dixon (1994). Although this is becoming more and more popular, there are still many optimization techniques in which finite-differencing is used to approximate the derivatives. In almost every NLP code, such finite-difference schemes are implemented.

Finite-difference schemes have been applied also to problems with stochastic functions. Kiefer and Wolfowitz (1952) were the first to describe the so-called stochastic (quasi) gradients; see also Blum (1954). Methods based on stochastic quasigradients are still subject of much research; for an overview, see Ermoliev (1988). So, although finite-difference schemes originate from obtaining gradient estimations for deterministic functions, they are applied also to stochastic functions.

Also in the field of design of experiments (DoE), schemes are available for obtaining gradient estimations. Some popular schemes are full or fractional factorial schemes, including the Plackett-Burman schemes. Contrary to finite-differencing, these schemes take noise into account. The schemes are such that, for example, the variance of the estimators is as small as possible. However, most DoE schemes assume a special form of the underlying model (e.g., polynomial) and lack of fit is usually not taken into account.

In Donohue *et al.* (1993) and Donohue *et al.* (1995), lack of fit is also taken into account besides the noise. These papers analyze what happens in a response surface methodology environment, when the postulated linear (resp. quadratic) model is misspecified, due to the true model structure being of second (resp. third) order. In these two papers new DoE schemes are derived by minimizing the integrated mean-squared error for either the predictor or the gradient. We think that, although valuable in an RSM environment, such estimations are less valuable for optimization purposes, since the integrated mean-squared error is not necessarily a good measure for the gradient in the current point. Moreover, the underlying assumption is that the real model is quadratic in Donohue *et al.* (1993) or third-order in Donohue *et al.* (1995). These very strict assumptions are not true in many cases. Finally, the results of the above mentioned papers suggest that, given the experimental region of interest, the points of the scheme shoul be placed as far as possible from the center. As is also pointed out in Safizadeh (2002), this is not logical and practically speaking not usefull.

The remainder of this paper is organized as follows. In Section 2.2, we analyze three

finite-difference schemes for obtaining gradient estimations. In Section 2.3, we do the same for two DoE schemes. In Section 2.4, we compare the errors of the five schemes. In Section 2.5, we deal with practical issues. We end with conclusions in Section 2.6.

2.2 Gradient estimation using finite-differencing

2.2.1 Forward finite-differencing

A classical approach to estimate the gradient of f is to apply forward finite-differencing (FFD) to the approximating function g, defined in (2.1). In this scheme, an estimator of the partial derivative, $\frac{\partial f(x)}{\partial x_i}$ $(i = 1, \ldots, n)$, is obtained by

$$\hat{\beta}_i^{\text{FFD}}(h) = \frac{g(x + he_i) - g(x)}{h}, \qquad h > 0, \tag{2.2}$$

where h is the stepsize and e_i is the ith unit vector. Using (2.1) and the Taylor formula, we can rewrite the estimator as

$$\begin{aligned} \hat{\beta}_i^{\text{FFD}} &= \frac{f(x + he_i) - f(x) + \varepsilon(x + he_i) - \varepsilon(x)}{h} \\ &= \frac{\partial f(x)}{\partial x_i} + \frac{1}{2} h e_i^T \nabla^2 f(x + \zeta h e_i) e_i + \frac{\varepsilon(x + he_i) - \varepsilon(x)}{h}, \end{aligned} \tag{2.3}$$

in which $0 \leq \zeta \leq 1$. We are interested now in how good this estimator is. Note that

$$E[\hat{\beta}_i^{\text{FFD}}] = \frac{\partial f(x)}{\partial x_i} + O(h), \qquad \text{Var}[\hat{\beta}_i^{\text{FFD}}] = \frac{2\sigma^2}{h^2}. \tag{2.4}$$

The estimators $\hat{\beta}_i^{\text{FFD}}$ and $\hat{\beta}_j^{\text{FFD}}$ are correlated, because both depend on $\varepsilon(x)$,

$$\begin{aligned} &\text{Cov}[\hat{\beta}_i^{\text{FFD}}, \hat{\beta}_j^{\text{FFD}}] = E\left((\hat{\beta}_i^{\text{FFD}} - E[\hat{\beta}_i^{\text{FFD}}])(\hat{\beta}_j^{\text{FFD}} - E[\hat{\beta}_j^{\text{FFD}}])\right) \\ &= \frac{1}{h^2} E\left((\varepsilon(x + he_i) - \varepsilon(x))(\varepsilon(x + he_j) - \varepsilon(x))\right) \\ &= \frac{1}{h^2} E\left(\varepsilon(x)^2\right) = \frac{\sigma^2}{h^2}, \qquad i \neq j. \end{aligned}$$

We are not only interested in the errors of the individual derivatives, but more in the error made in the resulting estimated gradient. A logical measure for the quality of our gradient estimation is the mean-squared error

$$E\left(\left\|\hat{\beta}^{\text{FFD}} - \nabla f(x)\right\|^2\right).$$

Not only the expectation is important, but also the variance

$$\text{Var}\left(\left\|\hat{\beta}^{\text{FFD}} - \nabla f(x)\right\|^2\right),$$

since high variance means that we run the risk that the error in a real situation might be much higher or lower than expected. Suppose for example that two simulation schemes have the same expected mean-squared error, then we prefer the scheme with the lowest variance. The variance can be used also in determining the optimal stepsize h, as we will see in Section 2.4. By defining the deterministic error

$$\text{error}_d^{\text{FFD}} = \left(\frac{f(x+he_1)-f(x)}{h}, \ldots, \frac{f(x+he_n)-f(x)}{h} \right)^T - \nabla f(x)$$

and the stochastic error

$$\text{error}_s^{\text{FFD}} = \left(\frac{\varepsilon(x+he_1)-\varepsilon(x)}{h}, \ldots, \frac{\varepsilon(x+he_n)-\varepsilon(x)}{h} \right)^T$$

we get

$$E\left(\left\| \hat{\beta}^{\text{FFD}} - \nabla f(x) \right\|^2 \right) = \left\| \text{error}_d^{\text{FFD}} \right\|^2 + E\left(\left\| \text{error}_s^{\text{FFD}} \right\|^2 \right).$$

From (2.3) we easily derive that

$$\left\| \text{error}_d^{\text{FFD}} \right\|^2 \leq \frac{1}{4} n h^2 D_2^2,$$

in which D_2 is the maximal second-order derivative of $f(x)$. Let us now analyze the stochastic error. The first part of the following theorem is well-known in the literature; see Zazanis and Suri (1993).

Theorem 2.1 *For FFD we have*

$$\begin{aligned}
&E\left(\left\| error_s^{FFD} \right\|^2 \right) = \frac{2n\sigma^2}{h^2} \\
&Var\left(\left\| error_s^{FFD} \right\|^2 \right) = \frac{n}{h^4} \left[n(M_4 - \sigma^4) + M_4 + 3\sigma^4 \right] \\
&Var\left(\left\| \hat{\beta}^{FFD} - \nabla f(x) \right\|^2 \right) \leq Var\left(\left\| error_s^{FFD} \right\|^2 \right) + 2n\sigma^2 D_2^2
\end{aligned}$$

in which M_4 is the fourth moment of $\varepsilon(x)$ in (2.1), i.e. $M_4 = E(\varepsilon(x)^4)$.

Proof By defining $\varepsilon_i = \varepsilon(x + he_i)$, $i = 1, ..., n$, and $\varepsilon_0 = \varepsilon(x)$, we have

$$E(\left\| \text{error}_s^{\text{FFD}} \right\|^2) = \frac{1}{h^2} E\left(\sum_i (\varepsilon_i - \varepsilon_0)^2 \right) = \frac{2n\sigma^2}{h^2}, \tag{2.5}$$

which proves the first part of the theorem. Considering the second part, we have

$$\text{Var}(\left\| \text{error}_s^{\text{FFD}} \right\|^2) = E(\left\| \text{error}_s^{\text{FFD}} \right\|^4) - E^2(\left\| \text{error}_s^{\text{FFD}} \right\|^2). \tag{2.6}$$

Let us now concentrate on the first term of the right-hand side of (2.6):

$$\begin{aligned} E(\left\|\text{error}_s^{\text{FFD}}\right\|^4) &= \frac{1}{h^4} E\left[\sum_i (\varepsilon_i^2 + \varepsilon_0^2 - 2\varepsilon_i\varepsilon_0) \sum_j (\varepsilon_j^2 + \varepsilon_0^2 - 2\varepsilon_j\varepsilon_0)\right] \\ &= \frac{1}{h^4}[n^2(M_4 + 3\sigma^4) + n(M_4 + 3\sigma^4)]. \end{aligned}$$

Substituting this result and the square of (2.5) into (2.6), we have the second part of the theorem. To prove the third part, first observe that

$$\begin{aligned} &\text{Var}\left(\left\|\hat{\beta}^{\text{FFD}} - \nabla f(x)\right\|^2\right) = \text{Var}(\left\|\text{error}_d^{\text{FFD}} + \text{error}_s^{\text{FFD}}\right\|^2) \\ &= E(\left\|\text{error}_d^{\text{FFD}} + \text{error}_s^{\text{FFD}}\right\|^4) - E^2(\left\|\text{error}_d^{\text{FFD}} + \text{error}_s^{\text{FFD}}\right\|^2) \\ &= \text{Var}(\left\|\text{error}_s^{\text{FFD}}\right\|^2) + 4\sum (\text{error}_d^{\text{FFD}})_i^2 E(\text{error}_s^{\text{FFD}})_i^2 \\ &\quad + 4\sum (\text{error}_d^{\text{FFD}})_i E(\text{error}_s^{\text{FFD}})_i^3. \end{aligned} \tag{2.7}$$

Further note that

$$\sum (\text{error}_d^{\text{FFD}})_i E(\text{error}_s^{\text{FFD}})_i^3 = 0,$$

and that

$$\sum (\text{error}_d^{\text{FFD}})_i^2 E(\text{error}_s^{\text{FFD}})_i^2 \leq n\left(\frac{1}{4}h^2 D_2^2\right)\left(\frac{2\sigma^2}{h^2}\right) = \frac{1}{2}n\sigma^2 D_2^2.$$

The third part of the theorem follows by substitution of the above results into (2.7). □

2.2.2 Central finite-differencing

A variant of the forward finite-differencing (FFD) approach is the central finite-differencing (CFD) approach. In this scheme, an estimation of the partial derivative $\frac{\partial f(x)}{\partial x_i}$, $i = 1, \ldots, n$, is obtained by

$$\hat{\beta}_i^{\text{CFD}}(h) = \frac{g(x + he_i) - g(x - he_i)}{2h}, \qquad h > 0, \tag{2.8}$$

where h is the stepsize and e_i is the ith unit vector. Using (2.1) we can rewrite the estimate as

$$\begin{aligned} \hat{\beta}_i^{\text{CFD}} &= \frac{f(x + he_i) - f(x - he_i) + \varepsilon(x + he_i) - \varepsilon(x - he_i)}{2h} \\ &= \frac{\partial f(x)}{\partial x_i} + \frac{h^2}{12}\nabla^3 f(x + \zeta_1 he_i)[e_i, e_i, e_i] + \frac{h^2}{12}\nabla^3 f(x + \zeta_2 he_i)[e_i, e_i, e_i] \\ &\quad + \frac{\varepsilon(x + he_i) - \varepsilon(x - he_i)}{2h}, \end{aligned} \tag{2.9}$$

where the last equality follows from the Taylor formulas

$$f(x+he_i) = f(x) + h\frac{\partial f(x)}{\partial x_i} + \frac{1}{2}h^2 e_i^T \nabla^2 f(x) e_i + \frac{h^3}{6}\nabla^3 f(x+\zeta_1 h e_i)[e_i, e_i, e_i]$$

in which $0 \leq \zeta_1 \leq 1$, and

$$f(x-he_i) = f(x) - h\frac{\partial f(x)}{\partial x_i} + \frac{1}{2}h^2 e_i^T \nabla^2 f(x) e_i - \frac{h^3}{6}\nabla^3 f(x+\zeta_2 h e_i)[e_i, e_i, e_i]$$

in which $0 \leq \zeta_2 \leq 1$. First note that

$$E[\hat{\beta}_i^{\mathrm{CFD}}] = \frac{\partial f(x)}{\partial x_i} + O(h^2), \qquad \mathrm{Var}[\hat{\beta}_i^{\mathrm{CFD}}] = \frac{\sigma^2}{2h^2}.$$

Contrary to the FFD estimations, $\hat{\beta}_i^{\mathrm{CFD}}$ and $\hat{\beta}_{j,i\neq j}^{\mathrm{CFD}}$, are not correlated. Indeed,

$$\begin{aligned}
\mathrm{Cov}[\hat{\beta}_i^{\mathrm{CFD}}, \hat{\beta}_j^{\mathrm{CFD}}] &= E\left((\hat{\beta}_i^{\mathrm{CFD}} - E[\hat{\beta}_i^{\mathrm{CFD}}])(\hat{\beta}_j^{\mathrm{CFD}} - E[\hat{\beta}_j^{\mathrm{CFD}}])\right) \\
&= \frac{1}{h^2}E[(\varepsilon(x+he_i) - \varepsilon(x-he_i))(\varepsilon(x+he_j) - \varepsilon(x-he_j))] \\
&= 0.
\end{aligned}$$

We analyze now the mean-squared error criterion

$$E\left(\left\|\hat{\beta}^{\mathrm{CFD}} - \nabla f(x)\right\|^2\right),$$

and its variance

$$\mathrm{Var}\left(\left\|\hat{\beta}^{\mathrm{CFD}} - \nabla f(x)\right\|^2\right).$$

By defining

$$\mathrm{error}_d^{\mathrm{CFD}} = \left(\frac{f(x+he_1) - f(x-he_1)}{2h}, \ldots, \frac{f(x+he_n) - f(x-he_n)}{2h}\right)^T - \nabla f(x)$$

and

$$\mathrm{error}_s^{\mathrm{CFD}} = \left(\frac{\varepsilon(x+he_1) - \varepsilon(x-he_1)}{2h}, \ldots, \frac{\varepsilon(x+he_n) - \varepsilon(x-he_n)}{2h}\right)^T,$$

we get

$$E\left(\left\|\hat{\beta}^{\mathrm{CFD}} - \nabla f(x)\right\|^2\right) = \left\|\mathrm{error}_d^{\mathrm{CFD}}\right\|^2 + E\left(\left\|\mathrm{error}_s^{\mathrm{CFD}}\right\|^2\right).$$

From (2.9), it is easy to verify that $\left\|\mathrm{error}_d^{\mathrm{CFD}}\right\|^2 \leq \frac{1}{36}nh^4 D_3^2$, in which D_3 is the maximal third-order derivative of $f(x)$. Let us now analyze the stochastic error. The first part of the following theorem is well-known in the literature; see Montgomery (1984).

Theorem 2.2 *For CFD we have*

$$E\left(\left\|error_s^{CFD}\right\|^2\right) = \frac{n\sigma^2}{2h^2},$$
$$Var\left(\left\|error_s^{CFD}\right\|^2\right) = \frac{n}{8h^4}\left[M_4 + \sigma^4\right],$$
$$Var\left(\left\|\hat{\beta}^{CFD} - \nabla f(x)\right\|^2\right) \leq Var(\left\|error_s^{CFD}\right\|^2) + \frac{1}{18}nh^2\sigma^2 D_3^2.$$

Proof By defining $\varepsilon_i = \varepsilon(x + he_i)$, $\varepsilon_{-i} = \varepsilon(x - he_i)$, $i = 1, ..., n$, and $\varepsilon_0 = \varepsilon(x)$ we have

$$E(\left\|\text{error}_s^{\text{CFD}}\right\|^2) = \frac{1}{4h^2}E\left(\sum_i(\varepsilon_i - \varepsilon_{-i})^2\right) = \frac{n\sigma^2}{2h^2}, \tag{2.10}$$

which proves the first part of the theorem. For the variance, we have

$$\text{Var}(\left\|\text{error}_s^{\text{CFD}}\right\|^2) = E(\left\|\text{error}_s^{\text{CFD}}\right\|^4) - E^2(\left\|\text{error}_s^{\text{CFD}}\right\|^2). \tag{2.11}$$

Let us now concentrate on the first term of the right-hand side in (2.11)

$$\begin{aligned} E(\left\|\text{error}_s^{\text{CFD}}\right\|^4) &= \frac{1}{16h^4}E\sum_i(\varepsilon_i^2 + \varepsilon_{-i}^2 - 2\varepsilon_i\varepsilon_{-i})\sum_j(\varepsilon_j^2 + \varepsilon_{-j}^2 - 2\varepsilon_j\varepsilon_{-j}) \\ &= \frac{1}{8h^4}\left(nM_4 + n(2n+1)\sigma^4\right). \end{aligned}$$

Substitution of this result and the square of (2.10) into formula (2.11) proves the second part of the theorem. The last part of the theorem follows similarly as in the proof of the last part of the previous theorem,

$$\begin{aligned} \text{Var}\left(\left\|\hat{\beta}^{\text{CFD}} - \nabla f(x)\right\|^2\right) &= \text{Var}(\left\|\text{error}_s^{\text{CFD}}\right\|^2) \\ &\quad + 4\sum(\text{error}_d^{\text{CFD}})_i^2 E(\text{error}_s^{\text{CFD}})_i^2 + 4\sum(\text{error}_d^{\text{CFD}})_i E(\text{error}_s^{\text{CFD}})_i^3 \\ &\leq \text{Var}(\left\|\text{error}_s^{\text{CFD}}\right\|^2) + 4n(\frac{1}{36}h^4 D_3^2)(\frac{\sigma^2}{2h^2}) + 0 \\ &= \text{Var}(\left\|\text{error}_s^{\text{CFD}}\right\|^2) + \frac{1}{18}nh^2\sigma^2 D_3^2. \end{aligned}$$

□

The result of this theorem can be simply checked for a special case. Suppose that all $\varepsilon(x)$ are standard normally distributed. Then by normalizing the stochastic error through the variance, we know that

$$\frac{2h^2}{\sigma^2}\left\|\text{error}_s^{\text{CFD}}\right\|^2$$

is the sum of n squared stochastic normally distributed variables, since

$$\left(\text{error}_s^{\text{CFD}}\right)_i = \varepsilon_i - \varepsilon_{-i}$$

is normally distributed. Hence,

$$\frac{2h^2}{\sigma^2}\left\|\text{error}_s^{\text{CFD}}\right\|^2$$

is $\chi^2(n)$ distributed, with expectation n and variance $2n$. So, we get

$$E(\left\|\text{error}_s^{\text{CFD}}\right\|^2) = n\frac{\sigma^2}{2h^2},$$

which is exactly the result of the theorem. Furthermore,

$$\text{Var}(\left\|\text{error}_s^{\text{CFD}}\right\|^2) = 2n\frac{\sigma^4}{4h^4} = \frac{n\sigma^4}{2h^4},$$

which agrees with the theorem, since for a normal distribution we have

$$M_4 = 3\sigma^4.$$

2.2.3 Replicated central finite-differencing

To decrease the stochastic error, one can repeat central finite-differencing K times. We call this replicated central finite-differencing (RCFD). Of course the deterministic error will not change by doing replications. The next theorem shows the expectation and variance of the resulting stochastic error.

Theorem 2.3 *For RCFD we have:*

$$\begin{aligned}
&E\left(\left\|error_s^{RCFD}\right\|^2\right) = \frac{n\sigma^2}{2h^2K},\\
&Var\left(\left\|error_s^{RCFD}\right\|^2\right) = \frac{n}{8h^4K^3}\left[M_4 + (4K-3)\sigma^4\right],\\
&Var\left(\left\|\hat{\beta}^{RCFD} - \nabla f(x)\right\|^2\right) \leq Var(\left\|error_s^{RCFD}\right\|^2) + \frac{1}{18K}nh^2\sigma^2D_3^2.
\end{aligned}$$

Proof By defining $\varepsilon_{ik} = \varepsilon_k(x + he_i)$, $\varepsilon_{-i,k} = \varepsilon_k(x - he_i)$, $i = 1, ..., n$, $k = 1, ..., K$ and $\varepsilon_{0k} = \varepsilon_k(x)$, where k denotes the kth replicate, we have for RCFD

$$\begin{aligned}
E(\left\|\text{error}_s^{\text{RCFD}}\right\|^2) &= \frac{1}{4h^2K^2}E\left(\sum_i\left(\sum_k(\varepsilon_{ik} - \varepsilon_{-i,k})\right)^2\right)\\
&= \frac{n\sigma^2}{2h^2K},
\end{aligned} \tag{2.12}$$

which proves the first part of the theorem. For the variance, we have

$$\text{Var}(\left\|\text{error}_s^{\text{RCFD}}\right\|^2) = E(\left\|\text{error}_s^{\text{RCFD}}\right\|^4) - E^2(\left\|\text{error}_s^{\text{RCFD}}\right\|^2). \tag{2.13}$$

Let us now concentrate on the first term of the right-hand side in (2.13),

$$\begin{aligned} & E(\left\|\text{error}_s^{\text{RCFD}}\right\|^4) \\ &= \frac{1}{16h^4K^4} E\left(\sum_{i,k,l}(\varepsilon_{ik}\varepsilon_{il} - \varepsilon_{-i,k}\varepsilon_{i,l} + \varepsilon_{-i,k}\varepsilon_{-i,l} - \varepsilon_{i,k}\varepsilon_{-i,l})\right)^2 \\ &= \frac{1}{16h^4K^4}[2KnM_4 + (2K^2n(n-1) + 6nK(K-1) + 2K^2n^2 + \ 4K^2n)\sigma^4]. \end{aligned}$$

Substitution of this result and the square of formula (2.12) into formula (2.13) proves the second part of the theorem. Finally, the third part can be derived almost identically as in the proof of the previous theorem. □

2.3 Gradient estimation using DoE schemes

2.3.1 Plackett-Burman schemes

We analyze now Design of Experiments (DoE) schemes for estimating the gradient. Let us start with the Plackett-Burman scheme. Suppose that we have a set of vectors $d_k \in \mathbb{R}^n,\ k = 1, \ldots, N$, with $\|d_k\| = 1$ and that we observe $g(x + hd_k)$ for fixed $x \in \mathbb{R}^n$ and $h > 0$. Define the matrix

$$X := \begin{pmatrix} 1 & \ldots & 1 \\ hd_1 & \ldots & hd_N \end{pmatrix}^T. \tag{2.14}$$

Now, suppose that N, with $n + 1 \leq N \leq n + 4$, is a multiple of four. Then, the Plackett-Burman scheme can be written as

$$H = \begin{pmatrix} 1 & \ldots & 1 \\ p_1 & \ldots & p_N \end{pmatrix}^T,$$

where $p_k \in \{-1, 1\}^n$. This so-called Hadamard matrix has the property $H^TH = NI$, where I is the identity matrix. For more information, see Box *et al.* (1987) or Kiefer and Wolfowitz (1952).

Now, let the vectors d_k in (2.14) be defined by $d_k = \frac{p_k}{\sqrt{n}}$, $k = 1, \ldots, N$. It then follows that

$$X^TX = \text{diag}\left(N,\ \tfrac{h^2N}{n}, \ldots, \tfrac{h^2N}{n}\right).$$

The vector containing the function value of f at x and the gradient can be estimated by the OLS estimator

$$\begin{aligned} \begin{pmatrix} \hat{\beta}_0^{\text{PB}} \\ \hat{\beta}^{\text{PB}} \end{pmatrix} &= (X^TX)^{-1}X^T\left(g(x + hd_1), \ldots, g(x + hd_N)\right)^T \\ &= (X^TX)^{-1}X^T[(f(x + hd_1), \ldots, f(x + hd_N))^T \\ &\quad + \ (\varepsilon(x + hd_1), \ldots, \varepsilon(x + hd_N))^T]. \end{aligned}$$

First note that

$$E[\hat{\beta}_0^{\mathrm{PB}}] = f(x) + O(h^2), \quad V[\hat{\beta}_0^{\mathrm{PB}}] = \tfrac{1}{n+1}\sigma^2 \tag{2.15}$$

$$E[\hat{\beta}_i^{\mathrm{PB}}] = \tfrac{\partial f(x)}{\partial x_i} + O(\sqrt{n}h), \quad V[\hat{\beta}_i^{\mathrm{PB}}] = \tfrac{n\sigma^2}{(n+1)h^2}, \quad i = 1, \ldots, n. \tag{2.16}$$

Since the columns of X are orthogonal, we have $\mathrm{Cov}[\hat{\beta}_i^{\mathrm{PB}}, \hat{\beta}_j^{\mathrm{PB}}] = 0$, $i \neq j$. Now defining D as the X matrix excluding the first column and

$$\begin{aligned} \mathrm{error}_d^{\mathrm{PB}} &= \frac{n}{h^2 N} D^T \left(f(x + hd_1), \ldots, f(x + hd_N)\right)^T - \nabla f(x) \\ \mathrm{error}_s^{\mathrm{PB}} &= \frac{n}{h^2 N} D^T \left(\varepsilon(x + hd_1), \ldots, \varepsilon(x + hd_N)\right)^T, \end{aligned}$$

we have

$$E\left(\left\|\hat{\beta}^{\mathrm{PB}} - \nabla f(x)\right\|^2\right) = \left\|\mathrm{error}_d^{\mathrm{PB}}\right\|^2 + E\left(\left\|\mathrm{error}_s^{\mathrm{PB}}\right\|^2\right).$$

Let us now concentrate on the deterministic error. Using the Taylor formula,

$$f(x + hd_k) = f(x) + hd_k^T \nabla f(x) + \frac{h^2}{2} d_k^T \nabla^2 f(x + \zeta h d_k) d_k,$$

in which $0 \leq \zeta \leq 1$, it is easy to derive that $\left\|\mathrm{error}_d^{\mathrm{PB}}\right\|^2 \leq \frac{n^2 h^2 D_2^2}{4}$, in which D_2 is an overall upper bound for the second-order derivative. Concerning the expectation and the variance of the stochastic error, we have the following theorem.

Theorem 2.4 *For Plackett-Burman schemes, we have*

$$\begin{aligned} &E\left(\left\|error_s^{PB}\right\|^2\right) = \frac{n^2\sigma^2}{Nh^2}, \\ &Var\left(\left\|error_s^{PB}\right\|^2\right) = \frac{n^4}{N^3 h^4}\left(M_4 + (\frac{2N}{n} - 3)\sigma^4\right), \\ &Var\left(\left\|\hat{\beta}^{PB} - \nabla f(x)\right\|^2\right) \leq Var(\left\|error_s^{PB}\right\|^2) + \frac{n^3\sigma^2 D_2^2}{N}. \end{aligned}$$

Proof For the Plackett-Burman schemes we have

$$\mathrm{error}_s^{\mathrm{PB}} = \frac{n}{h^2 N} D^T \nu = \frac{\sqrt{n}}{hN} P^T \nu,$$

in which P is the H matrix excluding the first column and

$$\nu = (\varepsilon(x + hd_1) \ldots \varepsilon(x + hd_N))^T.$$

We can derive the following:

$$E(\left\|\mathrm{error}_s^{\mathrm{PB}}\right\|^2) = \frac{n}{h^2 N^2} E(\left\|P^T \nu\right\|^2) = \frac{n^2\sigma^2}{Nh^2}, \tag{2.17}$$

which proves the first part of the theorem. For the variance, we have

$$\mathrm{Var}(\left\|\mathrm{error}_s^{\mathrm{PB}}\right\|^2) = E(\left\|\mathrm{error}_s^{\mathrm{PB}}\right\|^4) - E^2(\left\|\mathrm{error}_s^{\mathrm{PB}}\right\|^2). \tag{2.18}$$

Let us now concentrate on the first term of the right-hand side of (2.18),

$$\begin{aligned}
E(\left\|\mathrm{error}_s^{\mathrm{PB}}\right\|^4) &= \frac{n^2}{h^4N^4}E\left(\sum_j\left(\sum_i(P_{ij}\varepsilon_i)\sum_k(P_{kj}\varepsilon_k)\right)\right)^2 \\
&= \frac{n^2}{h^4N^4}E\left(\sum_{i,j,k}P_{ij}P_{kj}\varepsilon_i\varepsilon_k\right)^2 \\
&= \frac{n^2}{h^4N^4}E\left(\sum_{i,j,k,r,s,t}P_{ij}P_{kj}P_{sr}P_{tr}\varepsilon_i\varepsilon_k\varepsilon_s\varepsilon_t\right) \\
&= \frac{n^2}{h^4N^4}[2E\left(\sum_{i,j,k\neq i,r}P_{ij}P_{kj}P_{ir}P_{kr}\varepsilon_i^2\varepsilon_k^2\right) \\
&\quad + E\left(\sum_{i,j,s\neq i,r}P_{ij}P_{ij}P_{sr}P_{sr}\varepsilon_i^2\varepsilon_s^2\right) + E\left(\sum_{i,j,r}P_{ij}P_{ij}P_{ir}P_{ir}\varepsilon_i^4\right)],
\end{aligned}$$

where the last equality holds since the expectations of the terms $\varepsilon_i\varepsilon_k\varepsilon_s\varepsilon_t$, $\varepsilon_i^3\varepsilon_k$ and $\varepsilon_i^2\varepsilon_k\varepsilon_s$ are zero. We concentrate now on the three terms in the last equality. For the first term, we have

$$\begin{aligned}
&E\left(\sum_{i,j,k\neq i,r}P_{ij}P_{kj}P_{ir}P_{kr}\varepsilon_i^2\varepsilon_k^2\right) = \\
&\quad\left(\sum_{i,j,r\neq j}P_{ij}P_{ir}\sum_{k\neq i,}P_{kj}P_{kr}\right)\sigma^4 + \left(\sum_{i,j}P_{ij}P_{ij}\sum_{k\neq i,}P_{kj}P_{kj}\right)\sigma^4 \\
&= \sum_{i,j,r\neq j}P_{ij}P_{ir}(-P_{ij}P_{ir}) + n(N-1)\sigma^4 \\
&= nN(N-n)\sigma^4.
\end{aligned}$$

For the second term, it holds that

$$E\left(\sum_{i,j,s\neq i,r}P_{ij}P_{ij}P_{sr}P_{sr}\varepsilon_i^2\varepsilon_s^2\right) = n^2N(N-1)\sigma^4.$$

For the third term, we have

$$E\left(\sum_{i,j,r}P_{ij}P_{ij}P_{ir}P_{ir}\varepsilon_i^4\right) = n^2NM_4.$$

Substituting these results and the square of (2.17) into (2.18), we have proved the second part of the theorem. The third part of the theorem follows similarly as in the proof of the last part of Theorem 2.1,

$$\begin{aligned}
\mathrm{Var}\left(\left\|\hat{\beta}^{\mathrm{PB}} - \nabla f(x)\right\|^2\right) &= \mathrm{Var}(\left\|\mathrm{error}_s^{\mathrm{BP}}\right\|^2) \\
&\quad + 4\sum(\mathrm{error}_d^{\mathrm{PB}})_i^2 E(\mathrm{error}_s^{\mathrm{PB}})_i^2 + 4\sum(\mathrm{error}_d^{\mathrm{PB}})_i E(\mathrm{error}_s^{\mathrm{PB}})_i^3 \\
&\leq \mathrm{Var}(\left\|\mathrm{error}_s^{\mathrm{PB}}\right\|^2) + 4n(\frac{1}{4}nh^2D_2^2)(\frac{n\sigma^2}{Nh^2}) + 0 \\
&= \mathrm{Var}(\left\|\mathrm{error}_s^{\mathrm{PB}}\right\|^2) + \frac{n^3\sigma^2D_2^2}{N}.
\end{aligned}$$

□

2.3.2 Factorial schemes of resolution IV

Factorial schemes are based on the same principle as Plackett-Burman schemes, but now $N = 2^n$ for full factorial schemes and $N = 2^{n-p}$, $p \leq n$, for fractional factorial schemes. In this section we only consider resolution IV schemes. For the deterministic error, we can derive a better bound than for Plackett-Burman schemes. Again, we have

$$\mathrm{error}_d^{\mathrm{FD}} = \frac{n}{h^2N}D^T\left(f(x+hd_1),\ldots,f(x+hd_N)\right)^T - \nabla f(x).$$

Now using the Taylor formula

$$\begin{aligned}
f(x+hd_k) &= f(x) + hd_k^T\nabla f(x) + \frac{h^2}{2}d_k^T\nabla^2 f(x)d_k + \\
&\quad \frac{h^3}{6}\nabla^3 f(x+\zeta hd_k)[d_k,d_k,d_k], \qquad (2.19)
\end{aligned}$$

in which $0 \leq \zeta \leq 1$, and using the fact that, for resolution IV schemes for each vector d_k there exists exactly one other vector d_j in the factorial design scheme such that $d_k = -d_j$, by adding these two vectors we obtain

$$|f(x+hd_k) - f(x+hd_j)| \leq 2hd_k^T\nabla f(x) + \frac{h^3}{3}D_3,$$

in which D_3 is an overall upper bound for the third-order derivative. Combining all $N/2$ pairs of vectors we get

$$\begin{aligned}
\left\|\mathrm{error}_d^{\mathrm{FD}}\right\|^2 &= \left\|\frac{n}{h^2N}D^T\left(f(x+hd_1),\ldots,f(x+hd_N)\right)^T - \nabla f(x)\right\|^2 \\
&\leq \frac{n^2h^4D_3^2}{36}.
\end{aligned}$$

Concerning the stochastic error, we can derive the following results.

Theorem 2.5 *For factorial designs, we have*

$$E\left(\left\|error_s^{FD}\right\|^2\right) = \frac{n^2\sigma^2}{Nh^2},$$
$$Var\left(\left\|error_s^{FD}\right\|^2\right) = \frac{n^4}{N^3h^4}\left(M_4 + (\frac{2N}{n} - 3)\sigma^4\right),$$
$$Var\left(\left\|\hat{\beta}^{FD} - \nabla f(x)\right\|^2\right) \leq Var(\left\|error_s^{FD}\right\|^2) + \frac{1}{9N}n^3h^2\sigma^2D_3^2.$$

Proof Concerning the first and second part, we can derive the same results as for Plackett-Burman designs in the same way. Therefore, we omit the proof of these parts. The third part of the theorem follows similarly as in the proof of Theorem 2.4,

$$\begin{aligned}
\mathrm{Var}\left(\left\|\hat{\beta}^{\mathrm{FD}} - \nabla f(x)\right\|^2\right) &= \mathrm{Var}(\left\|\mathrm{error}_s^{\mathrm{FD}}\right\|^2) \\
&\quad + 4\sum(\mathrm{error}_d^{\mathrm{FD}})_i^2 E(\mathrm{error}_s^{\mathrm{FD}})_i^2 + 4\sum(\mathrm{error}_d^{\mathrm{FD}})_i E(\mathrm{error}_s^{\mathrm{FD}})_i^3 \\
&\leq \mathrm{Var}(\left\|\mathrm{error}_s^{\mathrm{FD}}\right\|^2) + 4n(\frac{1}{36}nh^4D_3^2)(\frac{n\sigma^2}{Nh^2}) \\
&= \mathrm{Var}(\left\|\mathrm{error}_s^{\mathrm{FD}}\right\|^2) + \frac{1}{9N}n^3h^2\sigma^2D_3^2.
\end{aligned}$$

□

2.4 Comparison of the schemes

In the previous sections, we have derived both the deterministic and the stochastic estimation errors for several schemes; see Table 2.1. The deterministic errors are increasing in the stepsize h, while the stochastic errors are decreasing in h. The expressions for the total error are convex functions of h. It is straightforward to calculate the optimal stepsize for each scheme such that the total error is minimized. The results are mentioned in the third row of Table 2.1.

Of course, usually we do not know the values for σ, D_2 and D_3. However, for a practical problem, we might estimate these values by sampling; see also the next section. Moreover, these optimal stepsizes give some indication; e.g., the stepsizes are increasing in σ and decreasing in N, D_2, and D_3, consistently with our intuition.

From the literature we know that CFD gives a much lower deterministic error than FFD. Concerning the stochastic error, we see from the table that the CFD scheme is four times better than FFD. However, the number of evaluations is two times more. To save evaluations, we can use a Plackett-Burman design: its number of evaluations is similar to the FFD scheme, but the stochastic error is two times lower; however, the deterministic error is n times higher. Full or fractional factorial designs of resolution IV have a much lower deterministic error than Plackett-Burman schemes. The stochastic error is similar,

Scheme	Forward FD	Replicated CFD	Plackett-Burman ($K = 1$ for CFD)	Factorial Res. IV
#eval	$n+1$	$2nK$	$n+1 \leq N \leq n+4$	$N = 2^{n-p}$
$\|\text{error}_d\|^2$	$\frac{1}{4}nh^2D_2^2$	$\frac{1}{36}nh^4D_3^2$	$\frac{1}{4}n^2h^2D_2^2$	$\frac{1}{36}n^2h^4D_3^2$
$E(\|\text{error}_s\|^2)$	$\frac{2n\sigma^2}{h^2}$	$\frac{n\sigma^2}{2h^2K}$	$\frac{n^2\sigma^2}{Nh^2}$	$\frac{n^2\sigma^2}{Nh^2}$
opt. h_e	$\sqrt[4]{8\frac{\sigma^2}{D_2^2}}$	$\sqrt[6]{9\frac{\sigma^2}{KD_3^2}}$	$\sqrt[4]{4\frac{\sigma^2}{ND_2^2}}$	$\sqrt[6]{18\frac{\sigma^2}{ND_3^2}}$
$\text{Var}(\|\text{error}_s\|^2)$	$\frac{n}{h^4}[n(M_4-\sigma^4)+ M_4+3\sigma^4]$	$\frac{n}{8h^4K^3}\left[M_4+(4K-3)\sigma^4\right]$	$\frac{n^4}{N^3h^4}\left(M_4+(\frac{2N}{n}-3)\sigma^4\right)$	$\frac{n^4}{N^3h^4}\left(M_4+(\frac{2N}{n}-3)\sigma^4\right)$
$\text{Var}(\|\text{error}_d+ \text{error}_s\|^2)$	$\frac{n}{h^4}[n(M_4-\sigma^4)+ M_4+3\sigma^4]+2n\sigma^2D_2^2$	$\frac{n}{8h^4K^3}\left[M_4+(4K-3)\sigma^4\right]$ $+\frac{1}{18K}nh^2\sigma^2D_3^2$	$\frac{n^4}{N^3h^4}\left(M_4+(\frac{2N}{n}-3)\sigma^4\right)$ $+\frac{n^3\sigma^2D_2^2}{N}$	$\frac{n^4}{N^3h^4}\left(M_4+(\frac{2N}{n}-3)\sigma^4\right)$ $+\frac{1}{9N}n^3h^2\sigma^2D_3^2$
opt. h_v	–	$\sqrt[6]{\frac{9(M_4+(4K-3)\sigma^4)}{2K^2\sigma^2D_3^2}}$	–	$\sqrt[6]{\frac{18n(M_4+(\frac{2N}{n}-3)\sigma^4)}{N^2\sigma^2D_3^2}}$

Table 2.1: Overview of the result for finite-difference and DoE schemes.

but since the number of evaluations is higher than for a Plackett-Burman scheme, the stochastic error can be made much lower by increasing N. However, this results in more evaluations. Observe also that the deterministic errors for Plackett-Burman and factorial schemes are independent of the number of evaluations, N. For the factorial schemes, this means also that we can decrease the stochastic error by increasing N, without affecting the deterministic error. Concerning the variances of the stochastic errors, it appears that CFD, Plackett-Burman, and factorial schemes are much better than FFD. When comparing RCFD and factorial schemes, it appears that the results are similar, since for a good comparison we have to take $N = 2nK$. However, note that, in the case of numerical noise, RCFD is not applicable, since the replicates will lead to the same outcomes. For such cases, factorial schemes are useful.

In Table 2.1, we have listed the variance of the stochastic errors and the total errors. Note that in the calculations for the optimal stepsizes h_e in Table 2.1, the variances of the errors are not taken into account. One can also determine a different stepsize by e.g. minimizing the expected error plus a certain number times the standard deviation. It can be verified easily that this will increase the optimal stepsizes h. In the last row of Table 2.1, we have calculated the optimal stepsize such that the total variance is minimized. This calculation is not possible for FFD and Plackett-Burman since those variances are decreasing functions in h. The optimal stepsizes h_v for the other schemes resemble the corresponding h_e. For example, suppose that all $\varepsilon(x)$ are standard normally distributed, then it can easily be verified that $h_v = \sqrt[6]{2}h_e \approx 1.1h_e$, since then $M_4 = 3\sigma^4$. This means that the stepsize h_e which minimizes the total error equals approximately the stepsize which minimizes the upper bound for the variance of the error. This property is an advantage of the CFD, RCFD and factorial schemes above the FFD and Plackett-Burman schemes.

In this paper, we focus on the estimation of gradients. However, note that CFD, Plackett-Burman, and factorial schemes deliver also better estimations for the function value. These better estimations can be valuable also for NLP solvers.

Concerning the amount of work needed to calculate the gradient estimation, we emphasize that the estimations based on the DoE schemes need nN additions / subtractions and n multiplications, while the FFD and CFD schemes need n additions / subtractions and n multiplications and RCFD need nK additions / subtractions and n multiplications. So, the extra amount of work needed in DOE schemes is limited.

2.5 Practical aspects

To obtain the values for the optimal stepsize, one has to estimate the unknown constants σ and either D_2 or D_3. In this section, first we show that the estimated gradient is not very

sensitive with respect to these constants. This means that even poor estimates of these quantities do not affect the quality of the gradient too much. We describe also how to derive good estimates of these constants. Finally, we describe how the Plackett-Burman scheme is successfully used in a practical application.

To analyze the sensitivity of the estimated gradient with respect to the unknown constants, let us assume that our estimates for σ, D_2 and D_3 are $\hat{\sigma} = \kappa\sigma$, $\hat{D}_2 = \kappa D_2$ and $\hat{D}_3 = \kappa D_3$, with $\kappa > 0$, respectively. Moreover, let us define the relative error rMSE as the quotient of the MSE when the above mentioned estimates are used for one of the constants and the MSE when the right values for the constants are used. For example

$$rMSE_{\widehat{\sigma}}^{\mathrm{FFD}} = \frac{E\left(\left\|\hat{\beta}^{\mathrm{FFD}} - \nabla f(x)\right\|^2\right)_{\widehat{\sigma}}}{E\left(\left\|\hat{\beta}^{\mathrm{FFD}} - \nabla f(x)\right\|^2\right)_{\sigma}},$$

where the subindex $\widehat{\sigma}$ indicates that the estimated value $\widehat{\sigma}$ is used instead of σ. Similar definitions hold for other schemes and other estimated constants. The following theorem shows expressions for this relative error.

Theorem 2.6 *For the relative errors we have*

$$\begin{aligned}
rMSE_{\widehat{\sigma}}^{FFD} &= rMSE_{\widehat{D}_2}^{FFD} = rMSE_{\widehat{\sigma}}^{PB} = rMSE_{\widehat{D}_2}^{PB} = \frac{1}{2}(\kappa + \frac{1}{\kappa}),\\
rMSE_{\widehat{D}_3}^{CFD} &= rMSE_{\widehat{D}_3}^{RCFD} = rMSE_{\widehat{D}_3}^{FD} = \frac{1}{3}(2\kappa^{2/3} + \frac{1}{\kappa^{4/3}}),\\
rMSE_{\widehat{\sigma}}^{CFD} &= rMSE_{\widehat{\sigma}}^{RCFD} = rMSE_{\widehat{\sigma}}^{FD} = \frac{1}{3}(\kappa^{4/3} + \frac{2}{\kappa^{2/3}}).
\end{aligned}$$

Proof We show first the results for FFD when $\widehat{D}_2$ is used. It can be verified easily that

$$E\left(\left\|\hat{\beta}^{\mathrm{FFD}} - \nabla f(x)\right\|^2\right)_{D_2} = \sqrt{2}n\sigma D_2,$$

by substituting the optimal stepsize h_e into the expression for the mean-squared error. Moreover, to calculate

$$E\left(\left\|\hat{\beta}^{\mathrm{FFD}} - \nabla f(x)\right\|^2\right)_{\widehat{D}_2}$$

we substitute the estimated stepsize

$$\sqrt[4]{8\frac{\sigma^2}{\widehat{D}_2^2}} = \sqrt[4]{8\frac{\sigma^2}{\kappa^2 D_2^2}}$$

into the expression for the MSE. We obtain

$$\begin{aligned} E\left(\left\|\hat{\beta}^{\mathrm{FFD}} - \nabla f(x)\right\|^2\right)_{\widehat{D}_2} &= \frac{1}{4}n\sqrt{8}\frac{\sigma}{\widehat{D}_2}D_2^2 + \frac{2n\sigma^2}{\sqrt{8}\frac{\sigma}{\widehat{D}_2}} \\ &= \frac{1}{2}(\kappa + \frac{1}{\kappa})\sqrt{2}n\sigma D_2 \\ &= \frac{1}{2}(\kappa + \frac{1}{\kappa})E\left(\left\|\hat{\beta}^{\mathrm{FFD}} - \nabla f(x)\right\|^2\right)_{D_2}. \end{aligned}$$

This proves the first part. The other results are obtained in a similar way. □

In Figure 2.1, the relative MSE errors are shown. From this figure it is clear that the estimated gradients are not very sensitive with respect to the constants. For example, an error of 20% for D_2 results into an 1.5% increase of the MSE for FFD and an error of 50% for D_2 results into an 7% increase.

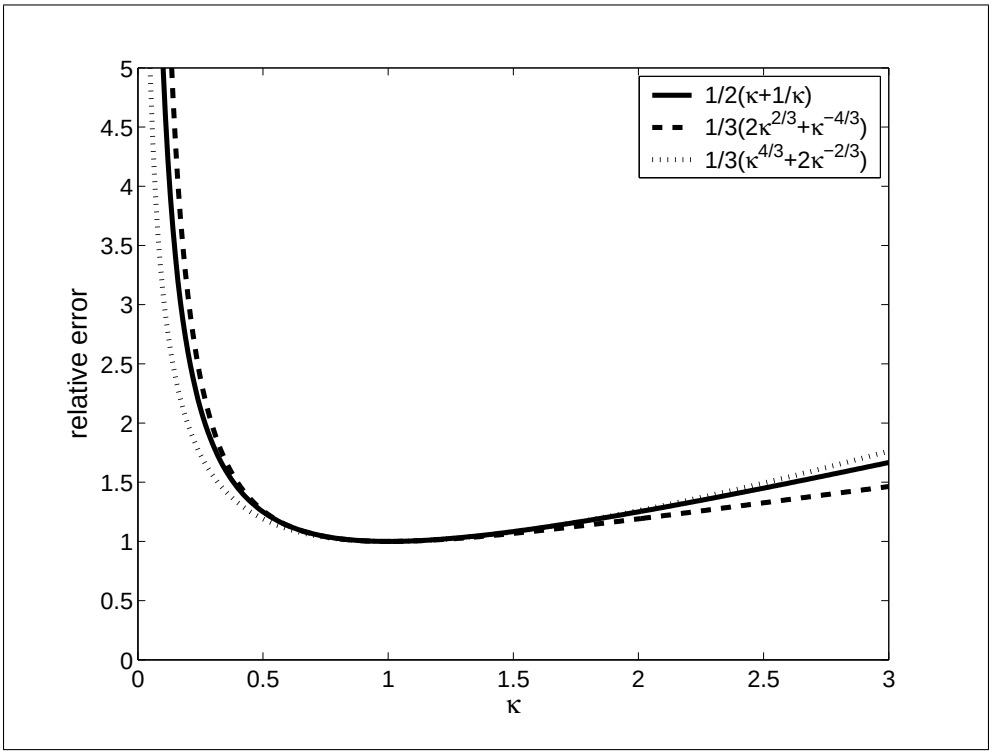

Figure 2.1: The relative errors.

To estimate the constants D_2 and D_3, we can use the same techniques as proposed for the classical finite-difference schemes. For example in Gill *et al.* (1981)[pp. 341 - 345], an algorithm is described in which D_2 is estimated by a second-order finite-difference scheme. Of course, as described in Gill *et al.* (1981), such an estimate is not made in each iteration, since this costs too many function evaluations, but only a few times. To estimate σ, we

can carry out replicates in a single point and use standard statistical methods. Again, such an estimate need not to be made in each iteration.

We have applied the Plackett-Burman scheme in the field of assets and liability management. The Cardano Risk Management Company has developed an assets and liabilities simulation model to analyze the financial performance of pension funds. Given the investment, premium, indexation strategy, and time horizon, this system can evaluate a number of performance indicators, like pension premium, premium reduction due to indexation and funding ratio. To evaluate these indicators, the system uses a large set of economic and liability scenarios. These scenarios are a realistic representation of all important developments of the future, which cannot be influenced by the pension fund. Normally speaking, the goal of a pension fund is to make the funding ratio as high as possible using only feasible values for the pension premium and the indexation reduction.

For the optimization problem under consideration, there were 11 optimization variables, representing the investment strategy. The objective is to maximize the average funding ratio over the entire time horizon. There are restrictions with respect to two other performance indicators: the pension premium and the indexation reduction. During the project, it appeared also that it was necessary to add a constraint on the so-called downside variation to avoid the Casino effect. The Casino effect means that the average funding ratio is increased by exploiting a small number of extreme scenarios, whereas the majority of the scenarios yields an unacceptably low funding ratio. Moreover, there are also restrictions on the investment strategy, e.g. on the running time, strike and hedge ratios. Note that, for one function evaluation, the system has to evaluate thousands of scenarios, due to the random exogenous parameters. This means that one function evaluation is time consuming.

Instead of using the classical forward finite-difference scheme, we used the Plackett-Burman scheme to obtain gradient estimates to feed the optimization method used in this project. For both schemes, the number of evaluations to estimate the gradient are almost the same. We compared the estimated gradients according to the two schemes. For the stepsize, we used a fixed percentage of the box region. To be more precise, we first scaled the box constraints to $0 \leq x \leq 1$ for the optimization variables x; then we used $h = 0.01$ as the stepsize for both the Plackett-Burman and forward finite-difference scheme. It appeared that the estimates obtained by the Plackett-Burman schemes are better than the forward finite-difference scheme. This is illustrated clearly in Figures 2.2 - 2.4, in which the derivative estimates according to both schemes for several output functions are given. In these figures, the graph for the function values are made by evaluating the function for several values and then connecting those points by a straight line.

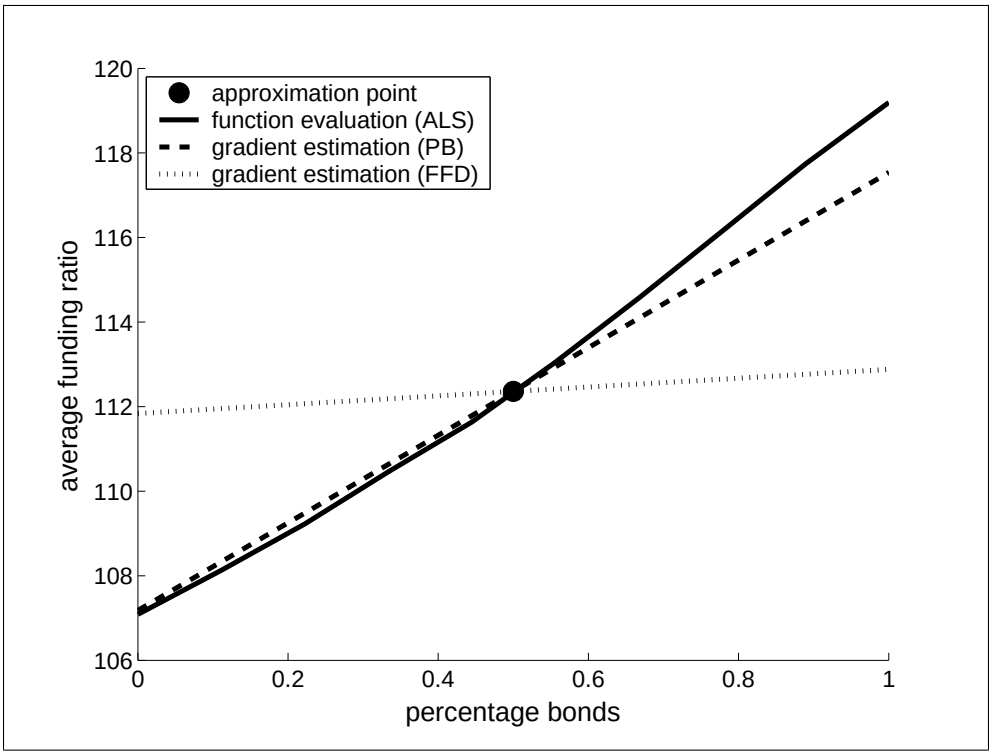

Figure 2.2: Comparison of PB and FFD estimations (1).

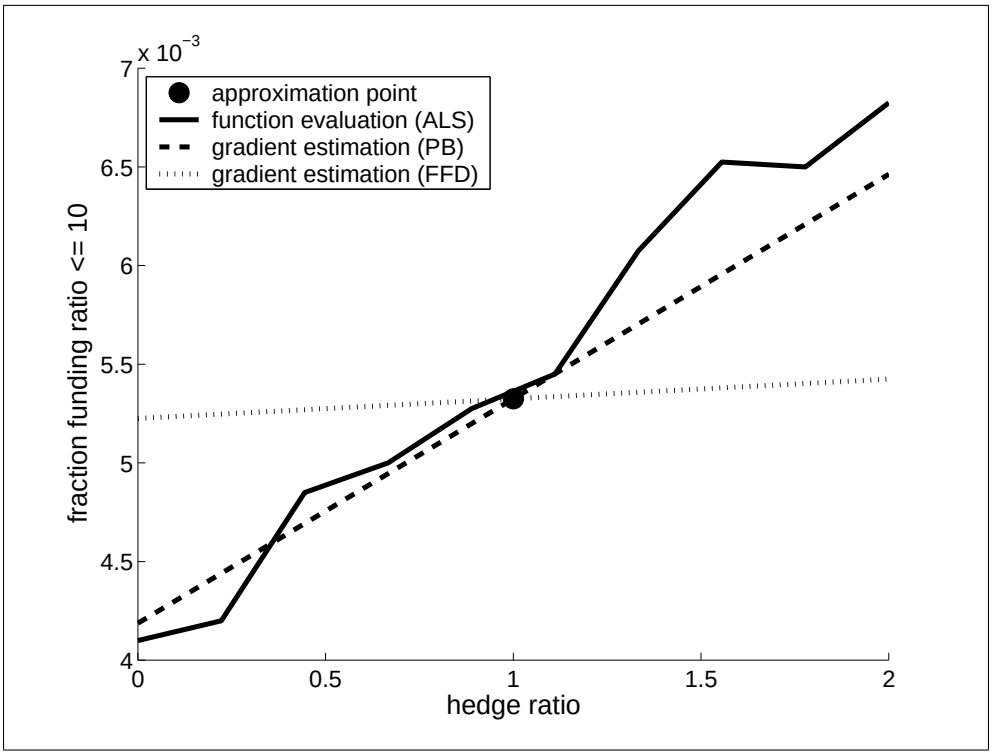

Figure 2.3: Comparison of PB and FFD estimations (2).

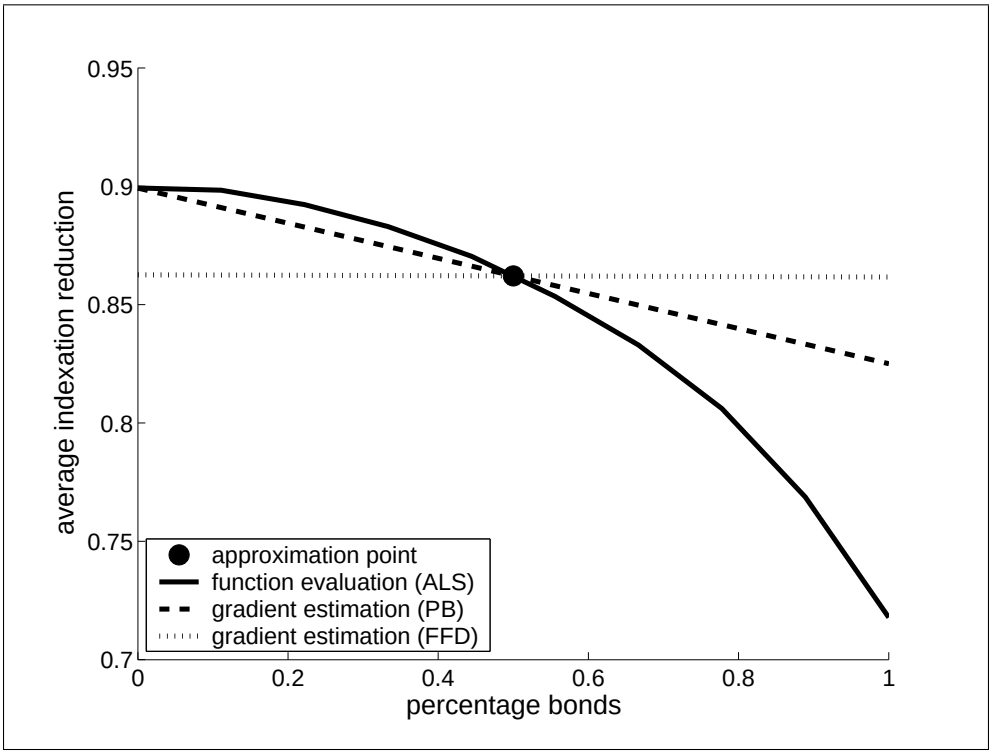

Figure 2.4: Comparison of PB and FFD estimations (3).

2.6 Conclusions

In the previous sections, we have discussed several methods for estimating the gradient of a function that is subject to i.i.d. random errors. The error that we make when estimating the gradient can be split into two parts: a deterministic error and a stochastic error. The deterministic error arises because we do not observe the function exactly at x, but in the neighborhood of x using finite stepsizes h. The stochastic error arises because of the noise. We have derived upper bounds for both the deterministic and stochastic errors, and we have discussed the advantages and disadvantages of three finite-difference schemes and two DoE schemes. We also discussed some practical issues.

The conclusion is that when the underlying function is indeed noisy, the fractional or full factorial DoE schemes are useful to reduce the stochastic error. Such schemes do not vary the variables one at a time, but vary all the variables simultaneously. The errors for factorial schemes are exactly the same as for replicated central finite-differences, but in case of numerical noise, we can use factorial schemes whereas replicates are meaningless. The Plackett-Burman schemes are useful when the evaluations are expensive. The stochastic errors of these schemes are two times lower than those of FFD, but the deterministic error is higher. Moreover, our error analysis indicates how to choose the stepsize h. It shows also that for CFD, RCFD and FD-schemes, the stepsize which minimizes the

total error, minimizes also the variance of the error. The DoE schemes can be included easily in the NLP solvers to estimate gradients.

We have shown also that the estimate gradients are not very sensitive to errors in the estimates for the constants σ, D_2 and D_3. Existing algorithms in the literature can be used to estimate these constants. Finally, we have shown the value of the Plackett-Burman scheme in a practical project in the field of assets and liabilities management.

Chapter 3

Gradient estimation using Lagrange interpolation polynomials

Abstract: In this paper we use Lagrange interpolation polynomials to obtain good gradient estimations. This is e.g. important for nonlinear programming solvers. As an error criterion we take the mean-squared error. This error can be split up into a deterministic and a stochastic error. We analyze these errors using (N times replicated) Lagrange interpolation polynomials. We show that the mean-squared error is of order $N^{-1+\frac{1}{2d}}$ if we replicate the Lagrange estimation procedure N times and use $2d$ evaluations in each replicate. As a result the order of the mean-squared error converges to N^{-1} if the number of evaluation points increases to infinity. Moreover, we show that our approach is also useful for deterministic functions in which numerical errors are involved. We also provide an optimal division between the number of grid points and replicates in case the number of evaluations is fixed. Further, it is shown that the estimation of the derivative is more robust when the number of evaluation points is increased. Finally, test results show the practical use of the proposed method.

3.1 Introduction

In this paper we estimate the gradient $\bigtriangledown f(x)$ of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. The function f is not explicitly known and we cannot observe it exactly. All observations are the result of an evaluation of the function, which is subject to certain perturbations. These perturbations can be of stochastic nature (e.g. in discrete-event simulation) or numerical nature (e.g. deterministic simulation models are often noisy due to numerical errors).

Obviously, gradients play an important role in all kind of optimization techniques. In most nonlinear programming (NLP) codes first-order and even second-order derivatives are used. Sometimes these derivatives can be calculated symbolically: becoming more and more popular is automatic differentiation, see e.g. Griewank (1989). Although this is becoming more and more popular, there are still many optimization solvers which use e.g. finite-differencing to obtain a good approximation of the gradient. See e.g. Gill *et al.* (1981) or Dennis and Schnabel (1989).

Finite-differences schemes have also been applied and analysed for problems with stochastic functions. Kiefer and Wolfowitz (1952) were the first to describe the so-called stochastic (quasi)gradients; see also Blum (1954). Methods based on stochastic quasi-

gradients are still subject of much research; for an overview see Ermoliev (1988). It was shown that the estimation error by using optimal stepsizes is $O(N^{-\frac{1}{2}})$ for forward finite-differencing and $O(N^{-\frac{2}{3}})$ for central finite-differencing, in which N is the number of replicates; see Glynn (1989), Zazanis and Suri (1993), L'Ecuyer and Perron (1994) and L'Ecuyer (1991). Moreover, Glynn (1989) developed a gradient estimator based on m evaluations instead of 2. He showed that for $m \to \infty$ the convergence rate is $\mathcal{L}_m N^{-1}$. However, this scheme appeared to be impractical since the constant $\mathcal{L}_m$ is highly increasing in m.

In this paper we will extend the finite-difference method. As in Glynn (1989), instead of using two evaluations for each dimension, we use more ($2d$) evaluations. We use Langrange interpolation polynomials to obtain a good point estimate of the gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$. More precisely, each partial derivative is estimated using an interpolating function $h(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{2d-1} x^{2d-1}$ that equals f in $2d$ evaluated points in one coordinate direction of f, with d a positive integer. Then $h'(0) = a_1$ is an estimate for this partial derivative. We consider the errors in the gradient estimation both due the deterministic approximation error ('lack of fit') and the presence of noise. We provide bounds for both the deterministic and the stochastic error. We show that the convergence rate is $N^{-1+\frac{1}{2d}}$, where N is the number of replicates of the Lagrange interpolation. This improves the above mentioned convergence rates for finite-differencing when $d \geq 2$. Note that for $d = 1$, the corresponding interpolation function $h(x)$ boils down into a linear Lagrange interpolation function, corresponds to the central finite-difference method. Compared with Glynn (1989), we observe that for $d \to \infty$ the convergence rate approaches $\mathcal{K}_d N^{-1}$, however, contrary to Glynn's method, our constants $\mathcal{K}_d$ are relatively small and bounded from above. Moreover, we provide some results in case we have a deterministic function in which numerical errors are involved. Given a fixed budget of evaluations, we provide an optimal division between the number of replicates (N) and the number of evaluations in such a replicate ($2d$). We also show that the estimation of the derivative is more robust against errors in the estimation of the parameters (variance, upper bound for the $(2d+1)$-th derivative), when the number of evaluation points is increased. The practical use of our method is shown by results on certain test problems.

This paper is organized as follows. Section 3.2 discusses the estimate of the gradient using Lagrange polynomials. The replicated Lagrange polynomials and the behavior of the mean-squared error are considered in Section 3.3. In Section 3.4 we consider the error of the gradient estimation if the function is deterministic. The optimal division between the number of replicates and the number of evaluations in such a replicate, if there is a fixed budget of evaluations, is discussed in Section 3.5. In Section 3.6 we show that the estimation is more robust when more evaluation points are used. Section 3.7 reports on the results of several test problems, and Section 3.8 concludes.

3.2 Gradient estimation of stochastic noisy functions

In this section we estimate the gradient of a $2d$ times continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ that is subject to stochastic noise using Lagrange interpolation polynomials. We provide an upper bound for the mean-squared error.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function subjected to stochastic noise. Hence, for a fixed $y \in \mathbb{R}^n$ we observe

$$g(y) = f(y) + \epsilon(y). \tag{3.1}$$

The error term $\epsilon(y)$ represents a random component. In this paper we assume that the error terms in (3.1) are i.i.d. random errors with $E[\epsilon(y)] = 0$ and $V[\epsilon(y)] = \sigma^2$. This assumption implies that the error terms do not depend on y. Note that g can also be a computer simulation model.

We will approximate $\frac{\partial f(y)}{\partial y_i}$, $i = 1, ..., n$, in a point $y \in \mathbb{R}^n$ using the approximation function g, defined in (3.1). Without loss of generality we take $y = (0, ..., 0)^T$. For convenience, let $I = \{-d, ..., -1, 1, ..., d\}$. Next, the function g is evaluated in the grid points $y_v^i = vhe_i$ for all $v \in I$, where $h > 0$ and e_i is the i-th unit vector of dimension n. Observe that the grid points are equidistant on each side of zero and that this distance is given by h (see Figure 3.1). Now, take the interpolating polynomial $h_i : \mathbb{R} \to \mathbb{R}$ defined as

$$h_i(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{2d-1} x^{2d-1} \tag{3.2}$$

that is exact in the evaluated points, i.e., according to (3.1) it holds that

$$h_i(x_v^i) = g(y_v^i) \text{ for all } v \in I, \tag{3.3}$$

where $x_v^i = e_i^T y_v^i$. Obviously, $h_i'(0) = a_1$ is an estimate of $\frac{\partial f(0)}{\partial y_i}$. This is illustrated in Figure 3.2. Using the Lagrange functions $l_{v,i} : \mathbb{R} \to \mathbb{R}$ defined as

$$l_{v,i}(x) = \Pi_{u \in I \setminus \{v\}} \frac{x - x_u^i}{x_v^i - x_u^i},$$

for any $v \in I$, (3.2) can be rewritten into

$$h_i(x) = \sum_{v \in I} l_{v,i}(x) g(y_v^i). \tag{3.4}$$

Hence, the derivative of $h_i(x)$ equals

$$h_i'(x) = \sum_{v \in I} \left[l_{v,i}(x) g(y_v^i) \sum_{u \in I \setminus \{v\}} \frac{1}{x - x_u^i} \right]. \tag{3.5}$$

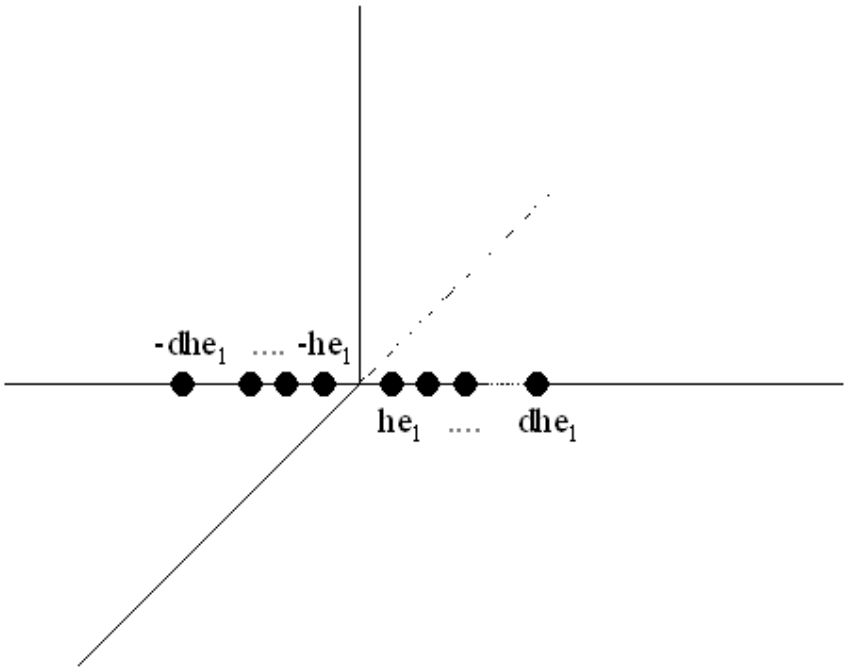

Figure 3.1: The $2d$ grid points for some h.

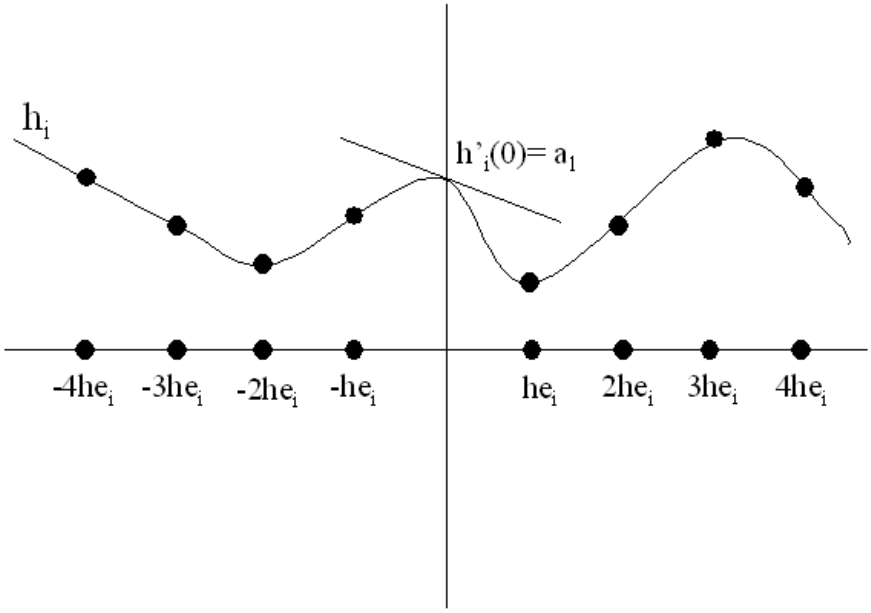

Figure 3.2: Estimate of gradient using interpolating polynomial.

d=1		d=2		d=3		d=4		d=5	
v	coeff $g(y_v^i)$	v	coeff $g(y_v^i)$	v	coeff $g(y_v^i)$	v	coeff $g(y_v^i)$	v	coeff $g(y_v^i)$
-1	-0.5	-2	0.0833	-3	-0.0167	-4	0.0036	-5	-0.0008
1	0.5	-1	-0.6667	-2	0.1500	-3	-0.0381	-4	0.0099
		1	0.6667	-1	-0.7500	-2	0.2000	-3	-0.0595
		2	-0.0833	1	0.7500	-1	-0.8000	-2	0.2381
				2	-0.1500	1	0.8000	-1	-0.8333
				3	0.0167	2	-0.2000	1	0.8333
						3	0.0381	2	-0.2381
						4	-0.0036	3	0.0595
								4	-0.0099
								5	0.0008

Table 3.1: Coefficients to generate estimate partial derivative.

From (3.5) it follows that the estimate of the partial derivative is a linear combination of the evaluations. Observe that the corresponding coefficients only depend on the $2d$ evaluation points. Table 3.1 provides the coefficients for $2d = 2, 4, 6, 8, 10$, respectively. The example in Section 3.6 will illustrate the use of the coefficients in Table 3.1.

Obviously, we are interested in the quality of $h_i'(0)$ as estimate of the partial derivative $\frac{\partial f(0)}{\partial y_i}$. Therefore we define

$$h'_{i,1}(x) = \sum_{v \in I} \left[l_{v,i}(x) f(y_v^i) \sum_{u \in I \setminus \{v\}} \frac{1}{x - x_u^i} \right] \tag{3.6}$$

and

$$h'_{i,2}(x) = \sum_{v \in I} \left[l_{v,i}(x) \epsilon(y_v^i) \sum_{u \in I \setminus \{v\}} \frac{1}{x - x_u^i} \right]. \tag{3.7}$$

It follows that

$$h_i'(x) = h'_{i,1}(x) + h'_{i,2}(x). \tag{3.8}$$

A well-known measure for the quality of the estimate of the partial derivative $\frac{\partial f(0)}{\partial y_i}$ by $h_i'(0)$ is the mean-squared error:

$$E\left(h_i'(0) - \frac{\partial f(0)}{\partial y_i} \right)^2.$$

By defining the deterministic error

$$\left(\text{error}_d^{h_i'} \right)^2 = \left(h'_{i,1}(0) - \frac{\partial f(0)}{\partial y_i} \right)^2$$

and the stochastic error

$$\left(\text{error}_s^{h_i'}\right)^2 = E(\ h'_{i,2}(0))^2$$

we get, because $E[\epsilon(x)] = 0$, that

$$E\left(\ h_i'(0) - \frac{\partial f(0)}{\partial y_i}\ \right)^2 = \left(\text{error}_d^{h_i'}\right)^2 + \left(\text{error}_s^{h_i'}\right)^2. \tag{3.9}$$

From (3.9) we learn that the mean-squared error is the sum of the deterministic and the stochastic error. The following Lemma provides an upper bound for the deterministic error.

Lemma 3.1 For the Lagrange estimate we have

$$\left(error_d^{h_i'}\right)^2 \leq M_{2d}^2 C_1^2(d) h^{4d-2},$$

where

$$C_1(d) = \frac{2}{(2d)!}\sum_{q=1}^{d}\left[q^{2d-1}\Pi_{r\in I\setminus\{q\}}\ \ \frac{\mid r\mid}{\mid r-q\mid}\right]$$

and M_{2d} is an upper bound for the $2d$ order derivative of f.

Proof For an upper bound of the deterministic error we use the Kowalewski's exact remainder for polynomial interpolation (cf. Davis (1975)[pp. 72]):

$$f_i(x) - h_{i,1}(x) = \frac{1}{(2d-1)!}\sum_{v\in I} l_{v,i}(x)\int_{x_v^i}^{x}(x_v^i - t)^{2d-1} f^{2d}(t)dt, \tag{3.10}$$

where f_i is the slice function of f taking the i^{th} component as variable. Taking the derivative to x on both sides of (3.10), substituting $x = 0$ and using
$\mid f^{2d}(y)\mid \leq M_{2d}$ we obtain

$$error_d^{h_i'} \leq \frac{M_{2d}}{(2d)!}\sum_{v\in I}\left[\mid l'_{v,i}(0)\mid (x_v^i - 0)^{2d}\right],$$

where $l'_{v,i}(0) = \Pi_{u\in I\setminus\{v\}}\left[\frac{0-x_u^i}{x_v^i - x_u^i}\right]\cdot\left[\sum_{u\in I\setminus\{v\}}\frac{1}{0-x_u^i}\right]$. Because

$$\mid l'_{v,i}(0)\mid = \left|\Pi_{u\in I\setminus\{v\}}\frac{0-x_u^i}{x_v^i - x_u^i}\right|\frac{1}{\mid x_v^i\mid}$$

and $x_u^i = hu$ for all $u \in I$, we have

$$\begin{aligned} error_d^{h_i'} &\leq \frac{M_{2d}}{(2d)!}\ 2\sum_{q=1}^{d}\left[(qh)^{2d}\Pi_{r\in I\setminus\{q\}}\frac{\mid r\mid h}{\mid r-q\mid h}\cdot\frac{1}{qh}\right] \\ &= M_{2d}C_1(d)h^{2d-1}, \end{aligned}$$

which completes the proof. □

The next Lemma shows, as illustrated in Figure 3.3, that $C_1(d)$ converges to zero. Hence, $error_d^{h_i'}$ will also converge to zero, if M_{2d} is bounded.

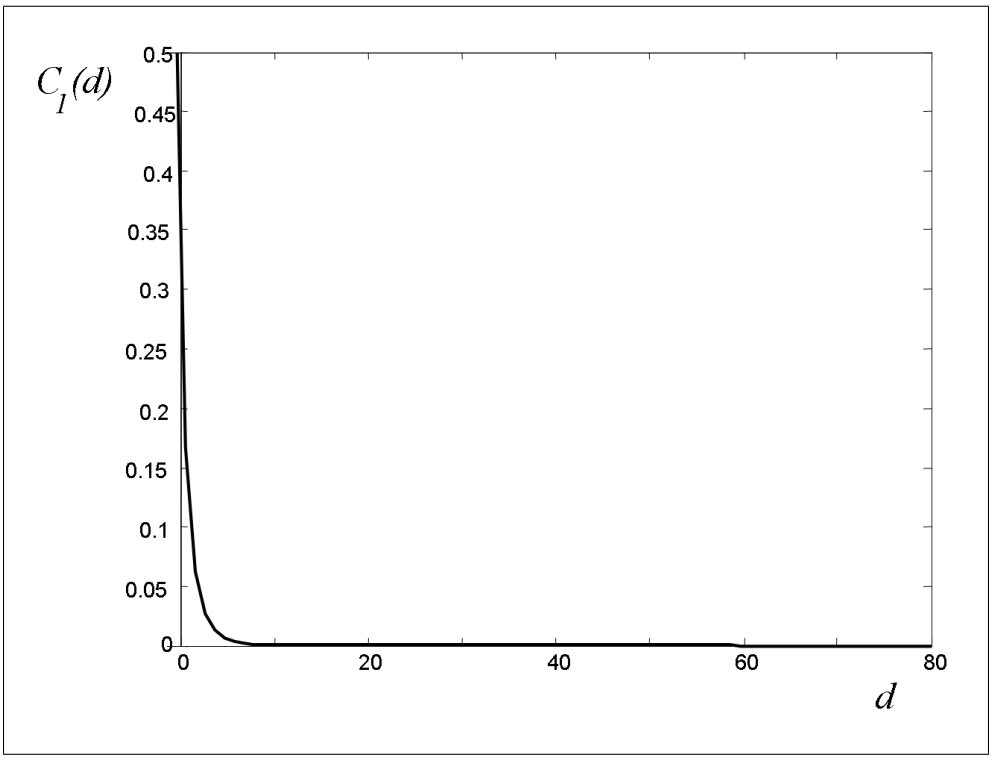

Figure 3.3: $C_1(d)$ converges to zero.

Lemma 3.2 Let $C_1(d) = \frac{2}{(2d)!}\sum_{q=1}^{d}\left[q^{2d-1}\Pi_{r\in I\setminus\{q\}}\ \ \frac{|r|}{|r-q|}\right]$. Then the following two statements hold:

$$(i)\ \ C_1(d) \leq 2d\left(\frac{3}{4-\epsilon}\right)^d,\ \text{with } \epsilon > 0 \text{ small},$$
$$(ii)\ \ C_1(d) \to 0 \text{ if } d \to \infty.$$

Proof It is sufficient to prove (i). First observe that $C_1(d)$ can be rewritten into

$$C_1(d) = \frac{2(d!)^2}{(2d)!}\sum_{q=1}^{2d-1}\frac{q^{2d-1}}{(d+q)!(d-q)!}.$$

Let

$$a_d = \frac{(2d)!}{2(d!)^2}.$$

Then

$$a_{d+1} = \frac{(d+1)^2}{(2d+2)(2d+1)} a_d.$$

Hence, there exists a small $\epsilon > 0$ such that $a_d \geq (4-\epsilon)a_{d+1}$ for large d. This implies that there is a constant c such that for large d we have

$$a_d \geq c(4-\epsilon)^d. \tag{3.11}$$

Let

$$b_d = \sum_{q=1}^{d} \frac{q^{2d-1}}{(d+q)!(d-q)!}.$$

Then for each $q = 1, ..., d$ we have

$$\begin{aligned}
\frac{q^{2d-1}}{(d+q)!(d-q)!} &\leq q^{-1} \frac{q^{2d-1}}{\left(\frac{d+q}{3}\right)^{d+q} \left(\frac{d-q}{3}\right)^{d-q}} \\
&= 3^{2d} \left(\frac{d+q}{q}\right)^{-d-q} \left(\frac{d-q}{q}\right)^{-d+q} \\
&= 3^{2d} \left[\left(\frac{1+x}{x}\right)^{-1-x} \left(\frac{1-x}{x}\right)^{-1+x}\right]^d \\
&\leq 3^{2d} \cdot \left(\frac{1}{3}\right)^d = 3^d
\end{aligned}$$

where the first inequality follows from Stirlings formula and that $q^{-1} \leq 1$. In the second inequality we use that the continuous and concave function $z : (0, 1] \to I\!R$ defined by

$$z(x) = \left(\frac{1+x}{x}\right)^{-1-x} \left(\frac{1-x}{x}\right)^{-1+x}$$

is upper bounded by $\frac{1}{3}$. Hence, we can conclude that

$$b_d \leq d3^d. \tag{3.12}$$

From (3.11) and (3.12) it follows that for large d we have

$$C_1(d) \leq 2d\left(\frac{3}{4-\epsilon}\right)^d,$$

which completes the proof. □

The following Lemma provides an expression for the stochastic error.

Lemma 3.3 For the Lagrange estimate we have

$$\left(error_s^{h_i'}\right)^2 = C_2(d)\frac{\sigma^2}{h^2}, \text{ with } C_2(d) = 4\sum_{q=1}^{d}\left(\Pi_{r\in I\setminus\{q\}} \quad \frac{\mid r \mid}{\mid r-q \mid}\frac{1}{q}\right)^2.$$

Proof We obtain

$$\begin{aligned}
\left(error_s^{h_i'}\right)^2 &= E(h'_{i,2}(0))^2 \\
&= E\left(\sum_{v\in I}\Pi_{u\in I\setminus\{v\}}\frac{0-x_u^i}{x_v^i-x_u^i}\epsilon(x_v^i)\sum_{u\in I\setminus\{v\}}\frac{1}{0-x_u^i}\right)^2 \\
&= \sigma^2\sum_{v\in I}\left(\Pi_{u\in I\setminus\{v\}}\frac{0-x_u^i}{x_v^i-x_u^i}\sum_{u\in I\setminus\{v\}}\frac{1}{0-x_u^i}\right)^2 \\
&= 4\frac{\sigma^2}{h^2}\sum_{q=1}^{d}\left(\Pi_{r\in I\setminus\{q\}} \quad \frac{\mid r \mid}{\mid r-q \mid}\frac{1}{q}\right)^2
\end{aligned}$$

which completes the proof. □

The next lemma shows, as Figure 3.4 suggests, that $C_2(d)$ is upper bounded.

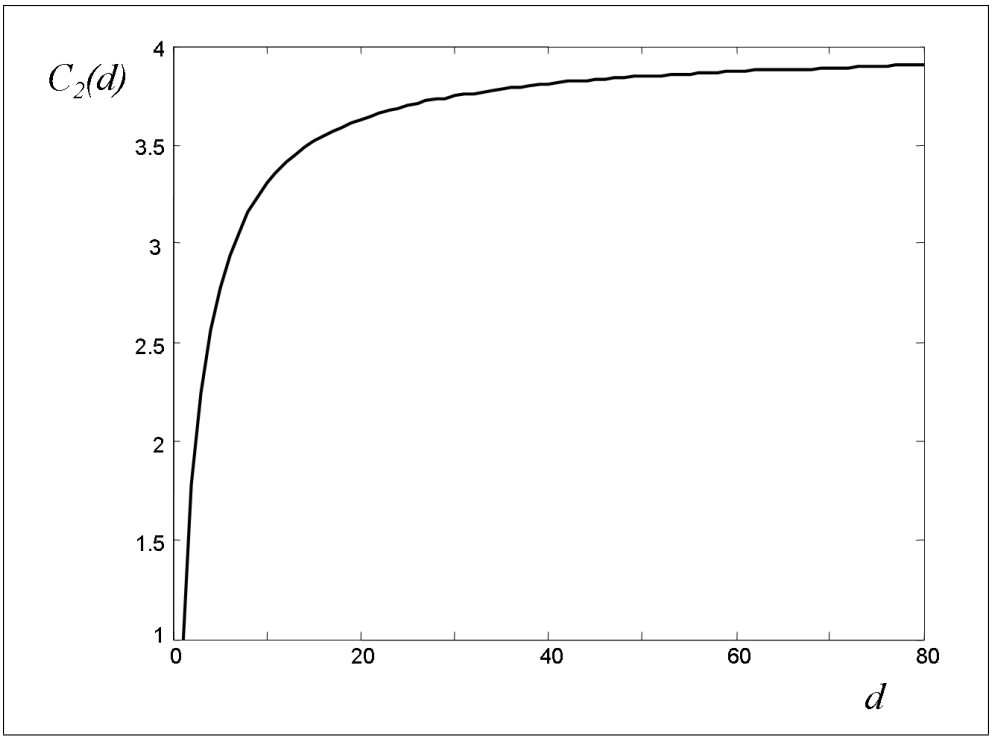

Figure 3.4: The behavior of $C_2(d)$ if the number of evaluation points increases.

Lemma 3.4 Let $C_2(d) = 4\sum_{q=1}^{d}\left(\Pi_{r\in I\setminus\{q\}} \ \frac{|r|}{|r-q|}\frac{1}{q}\right)^2$.
Then $C_2(d) \leq \frac{2}{3}\pi^2$ for all d.

Proof Observe that

$$C_2(d) = 4\sum_{q=1}^{2d-1}\big(\frac{(d!)^2}{(d+q)!(d-q)!)}\frac{1}{q}\big)^2.$$

Because $\frac{(d!)^2}{(d+q)!(d-q)!)} \leq 1$ for all q we have that

$$C_2(d) \leq 4\sum_{q=1}^{2d-1}\frac{1}{q^2} \leq 4\cdot\frac{1}{6}\pi^2 = \frac{2}{3}\pi^2,$$

which completes the proof. □

3.3 The use of replicates

In this section we estimate the gradient of a $2d$ continuous differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ that is subject to stochastic noise by replicating the Lagrange estimation of the previous sections. We investigate the mean-squared error.

The following lemmata with respect to the deterministic and stochastic error follow straightforward from Lemma 3.1 and Lemma 3.3, respectively. Obviously, the upper bound for the deterministic error will not change in case of replicates.

Lemma 3.5 For the Lagrange estimation with N replicates we have

$$\left(error_d^{h_i'}\right)^2 \leq M_{2d}^2 C_1^2(d) h^{4d-2}.$$

Evidently, the stochastic error in case of replicates is decreased by a factor N, the number of replicates.

Lemma 3.6 For the Lagrange estimation with N replicates we have

$$\left(error_s^{h_i'}\right)^2 = C_2(d)\frac{\sigma^2}{Nh^2}.$$

In the final part of this section we determine the stepsize h that minimizes the mean-squared error. From Lemma 3.5 and 3.6 it follows that the mean-squared error, as a function of h, is upper bounded by

$$UMSE(h) = M_{2d}^2 C_1^2(d) h^{4d-2} + C_2(d)\frac{\sigma^2}{Nh^2}. \tag{3.13}$$

The following Theorem states the optimal stepsize for the upper bound and shows that the minimum mean-squared error converges to N^{-1} if d goes to infinity.

d	$UMSE$
1	$1.00 M_2 \sigma N^{-\frac{1}{2}}$
2	$1.10 (M_4)^{\frac{1}{2}} \sigma^{\frac{3}{2}} N^{-\frac{3}{4}}$
10	$1.68 (M_{20})^{\frac{1}{10}} \sigma^{\frac{19}{10}} N^{-\frac{19}{20}}$
20	$1.94 (M_{40})^{\frac{1}{20}} \sigma^{\frac{39}{20}} N^{-\frac{39}{40}}$
50	$2.12 (M_{100})^{\frac{1}{50}} \sigma^{\frac{99}{50}} N^{-\frac{99}{100}}$

Table 3.2: The $UMSE$ for some values of d.

Theorem 3.1 *Let $UMSE(h)$ be defined as in (3.13). Then :*

(i) *The optimal step size is* $h^* = (PN)^{\frac{-1}{4d}}$,

$$P = \left(\frac{C_2(d)\sigma^2}{M_{2d}^2 C_1^2(d)(2d-1)} \right)^{-1},$$

(ii) *The minimum of* $UMSE$ *is*

$$UMSE(h^*)^{\frac{-1}{4d}} = M_{2d}^{\frac{1}{d}} \sigma^{2-\frac{1}{d}} C_3(d) N^{-1+\frac{1}{2d}}$$

$$\text{with } C_3(d) = (C_1(d))^{\frac{1}{d}} (C_2(d))^{1-\frac{1}{2d}} (2d-1)^{\frac{1}{2d}} \left(\frac{2d}{2d-1} \right),$$

(iii) $C_3(d) \leq 0.9\pi^2$ *for large* d,

(iv) $UMSE(h^*) \to \mathcal{O}(N^{-1})$ *if* $d \to \infty$.

Proof The proof of (i) and (ii) is straightforward and (iv) results from (ii) and (iii). We will prove (iii). From Lemma 3.2 (i) it follows that $C_1(d)^{\frac{1}{d}} = (2d)^{\frac{1}{d}} \left(\frac{3}{4-\epsilon}\right)$. Because $(2d)^{\frac{1}{d}}$ converges to 1, we have that $(2d)^{\frac{1}{d}} \leq 1.1$ for large d and $\left(\frac{3}{4-\epsilon}\right) \leq 1$. Hence,

$$C_1(d)^{\frac{1}{d}} \leq 1.1 \text{ if } d \text{ is large.} \tag{3.14}$$

Obviously, it holds that

$$C_2(d)^{1-\frac{1}{2d}} \leq \frac{2}{3}\pi^2. \tag{3.15}$$

Because both $(2d-1)^{\frac{1}{2d}}$ and $\frac{2d}{2d-1}$ converge to 1 we have that both terms are upper bounded by 1.1 if d is large. Combining this last observation with (3.14) and (3.15) we obtain

$$C_3(d) \leq 1.1 \cdot \frac{2}{3}\pi^2 \cdot 1.1 \cdot 1.1 < 0.9\pi^2.$$

□

In Figure 3.5 the behavior of $C_3(d)$ is illustrated. Table 3.2 provides the UMSE for some specific values of d. Observe that already for small d the best results in forward finite-differencing ($\mathcal{O}(N^{-\frac{1}{2}})$) and central finite-differencing ($\mathcal{O}(N^{-\frac{2}{3}})$) are improved. In

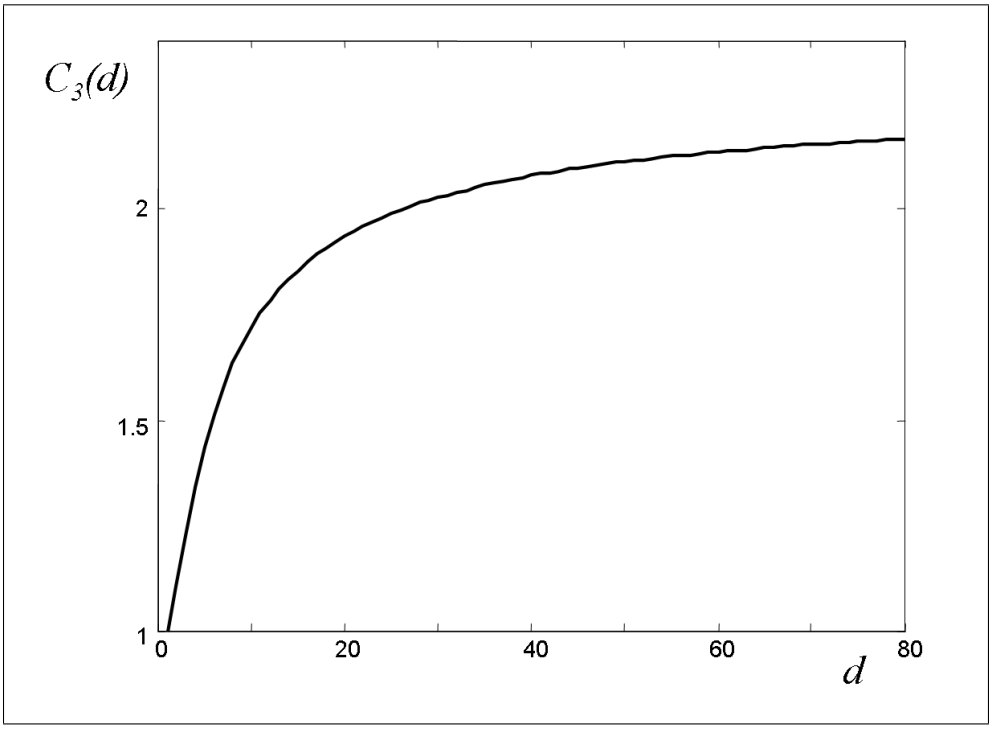

Figure 3.5: The behavior of $C_3(d)$.

fact, for $d = 1$ our result is identical to forward finite-differencing. It is interesting to compare the results in this table with the results of Glynn (1989). The order of convergence is the same, however, the constants explode for increasing d for his method. The corresponding constants for his method are e.g.: 31 for $d = 2$, 1.5×10^9 for $d = 10$ and 3.8×10^{20} for $d = 20$.

3.4 Gradient estimation of numerically noisy functions

In this section we estimate the gradient of a $2d$ times continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ that is subjected to numerical noise using Lagrange polynomials.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function that is subjected to numerical noise. Hence, for a fixed $y \in \mathbb{R}^n$ we observe

$$g(y) = f(y) + \epsilon(y),$$

where $\epsilon(y)$ is the fixed, unknown numerical error. To estimate the gradient of f we take the same approach as in Section 3.2. Let the function h, $h'_{i,1}$ and $h'_{i,2}$ be defined as in (3.4), (3.6)and (3.7), respectively.

Then the total error of the estimate of the partial derivative is equal to

$$\left|\frac{\partial f(0)}{\partial y_i} - h_i'(0)\right|. \tag{3.16}$$

We define the deterministic model error by

$$\left|\frac{\partial f(0)}{\partial y_i} - h_{i,1}'(0)\right|$$

and the numerical error by

$$\left|h_{i,2}'(0)\right|.$$

We get, by using (3.8), the following upper bound for the total error

$$\left|\frac{\partial f(0)}{\partial y_i} - h_i'(0)\right| \leq \left|\frac{\partial f(0)}{\partial y_i} - h_{i,1}'(0)\right| + \left|h_{i,2}'(0)\right|. \tag{3.17}$$

Similarly to Section 3.2 we can provide upper bounds for the deterministic model and the numerical errors. The proofs of the following two Lemmata are omitted because they are almost identical to the proofs of Lemma 3.1 and 3.3, respectively.

Lemma 3.7 For the Lagrange estimate we have

$$\left|\frac{\partial f(0)}{\partial y_i} - h_{i,1}'(0)\right| \leq M_{2d} C_1(d) h^{2d-1}.$$

Lemma 3.8 For the Lagrange estimate we have

$$\left|h_{i,2}'(0)\right| \leq C_2(d)^{\frac{1}{2}} \frac{K}{h},$$

where K is an upper bound of ϵ.

In the final part of this section we determine the stepsize h that minimizes the total error. From 3.7 and 3.8 it follows that the total error TE, as a function of h, is upper bounded by

$$UTE(h) = M_{2d} C_1(d) h^{2d-1} + C_2(d)^{\frac{1}{2}} \frac{K}{h}. \tag{3.18}$$

The next Theorem provides the stepsize that minimizes the total error.

Theorem 3.2

(*i*) *The optimal step size* h^* *is*

$$h^* = \left(\frac{C_2(d)^{\frac{1}{2}} K}{(2d-1)^{2d-1} M_{2d} C_1(d)}\right)^{-\frac{1}{2d}},$$

(*ii*) *The minimum of* UTE *is*

$$UTE(h^*) = M_{2d}^{1-\frac{1}{2d}} C_1(d)^{\frac{1}{2d}} C_2(d)^{\frac{1}{2}-\frac{1}{4d}} K^{1-\frac{1}{2d}} (2d-1)^{\frac{1}{2d}} \left(\frac{2d}{2d-1}\right).$$

$\sigma = 1$, $M_{2d} = 1$			
B	d	error	N
$B \leq 23$	1	0.2879	6
$B \leq 803$	2	0.0207	134
$B \leq 21984$	3	0.0013	2748
$B \leq 386720$	4	0.0001	38672
$B \leq 5461476$	5	9.85E -0.6	455123

$\sigma = 0.1$, $M_{2d} = 1$			
B	d	error	N
$B \leq 3$	1	0.0349	1
$B \leq 12$	2	0.0148	2
$B \leq 240$	3	0.0012	30
$B \leq 3880$	4	0.0001	388
$B \leq 54660$	5	9.85E -0.6	4555

$\sigma = 0.01$, $M_{2d} = 1$			
B	d	error	N
$B \leq 3$	1	0.0011	1
$B \leq 12$	2	0.0003	2
$B \leq 24$	3	0.0001	3
$B \leq 40$	4	0.0001	4
$B \leq 600$	5	9.03E -0.6	50

$\sigma = 010$, $M_{2d} = 1$			
B	d	error	N
$B \leq 2376$	1	0.2901	594
$B \leq 79932$	2	0.0208	13322

Table 3.3: The optimal division between d and N at a fixed number of evaluations B.

The proof is straightforward and is therefore omitted. Observe that for the special case $d = 1$ that the result in Theorem 3.2 is similar to the result obtained in Gill *et al.* (1981)[pp. 340], for the forward finite-difference approximation.

3.5 Grid points versus replicates

In this section we provide an optimal division between the number of grid points and replicates in case the number of evaluations is fixed. Let B be the total number of evaluations available, N the number of replicates and $2d$ the number of evaluations per replicate. The problem to solve is the following:

$$\begin{aligned} \text{minimize} \quad & UMSE(KN^{\frac{-1}{4d}}) = C_3(d) \left(\frac{M_{2d}}{\sigma}\right)^{\frac{1}{d}} \sigma^2 N^{-1+\frac{1}{2d}} \\ \text{s.t} \quad & B = 2dN, \\ & d, N \text{ positive integers.} \end{aligned} \tag{3.19}$$

In Table 3.3 we provide the optimal division between d and N for some values of B and a specific ratio of $\frac{M_{2d}}{\sigma}$. In the upper left cell of Table 3.3 we have chosen $\sigma = 1$ and $M_{2d} = 1$. This cell illustrates that for a fixed budget $B = 24$ till $B = 803$ it is optimal to evaluate 4 ($d = 2$), points in each replicate. Obviously, in this case the number of replicates is determined by the quotient of the budget and 4. From $B = 804$ till $B = 21983$ it turns out that it is optimal to evaluate 6 points in each replicate. For example, if $B = 6000$ then we take $d = 3$, which equals 6 evaluations, and 1000 replicates.

The other three cells of Table 3.3 present the results for different ratios of σ and M_{2d}. Observe that the error decreases if σ decreases. Moreover, the turning points to increase the number of grid points also decreases if σ decreases. For example, if $\sigma = 1$, then we turn to 6 grid points if $B = 804$, whereas if $\sigma = 0.01$ we already increase to 6 grid points if $B = 12$.

Observe that Table 3.3 suggests that $d = \mathcal{O}(\log(B))$ if $B \to \infty$. Indeed, this observation can be made plausible using the following arguments. There are two possible scenarios for the behavior of d if B becomes large. The first one is that $d \to \infty$. Because in this situation $C_3(d) \to c_3$. Then a good approximation for (3.19) is obtained if we solve the following relaxation:

$$\begin{aligned} \text{minimize} \quad & \left(\frac{M_{2d}}{\sigma}\right)^{\frac{1}{d}} N^{-1+\frac{1}{2d}} \\ \text{s.t.} \quad & B = 2dN. \end{aligned}$$

The minimum is found by determining the stationary points of

$$\log\left[\left(\frac{B}{2d}\right)^{-1+\frac{1}{2d}} \left(\frac{M_{2d}}{\sigma}\right)^{\frac{1}{d}}\right],$$

which boils down to

$$\frac{1}{2d^2}\left[-\log\left(\frac{B}{2d}\right) - 2\log\left(\frac{M_{2d}}{\sigma}\right) + 2d - 1\right] = 0,$$

which shows that $d \approx k\log(B)$ for some constant k. Substituting this result in (3.19) we obtain

$$C_3(d)\left(\frac{M_{2d}}{\sigma}\right)^{\frac{1}{d}} \sigma^2 \left(\frac{B}{2k\log(B)}\right)^{-1+\frac{1}{2k\log(B)}}. \tag{3.20}$$

Let $\beta(d) = C_3(d)\left(\frac{M_{2d}}{\sigma}\right)^{\frac{1}{d}} \sigma^2$ then (3.20) can be rewritten into

$$\beta(d)\left(\frac{B}{2k\log(B)}\right)^{-1+\frac{1}{2k\log(B)}}. \tag{3.21}$$

This would end the argumentation. However, it can be the case that $d \to d^*$. Then there exists a $\hat{d} \leq d^*$ such that the minimum of (3.19) is attained in $\hat{d}$ and equals

$$C_3(\hat{d})\left(\frac{M_{2\hat{d}}}{\sigma}\right)^{\frac{1}{\hat{d}}} \sigma^2 \left(\frac{B}{2\hat{d}}\right)^{-1+\frac{1}{2\hat{d}}}. \tag{3.22}$$

Let $\alpha(\hat{d}) = C_3(\hat{d})\left(\frac{M_{2\hat{d}}}{\sigma}\right)^{\frac{1}{\hat{d}}} \sigma^2$, then (3.22) is equal to

$$\alpha(\hat{d})\left(\frac{B}{2\hat{d}}\right)^{-1+\frac{1}{2\hat{d}}}. \tag{3.23}$$

Now, we can conclude that $d = \mathcal{O}(\log B)$ if the minimum of (3.19) is attained in the situation where $d \to \infty$. Hence, we are finished if we can show that the expression in (3.23) is larger than (3.21).

Observe that $\alpha(\hat{d})$ is a constant and $\beta(d)$ is bounded under the assumption that $M_{2d} \leq a^{2d}$ for some constant $a > 1$ and Theorem 3.1 (iii). Hence, it is sufficient to prove that

$$\frac{\left(\frac{B}{2k\log(B)}\right)^{-1+\frac{1}{2k\log(B)}}}{\left(\frac{B}{2\hat{d}}\right)^{-1+\frac{1}{2\hat{d}}}} \to 0 \quad \text{if} \quad B \to \infty \tag{3.24}$$

Straightforward calculations yield that (3.24) can be rewritten into $v(B)w(B)$ where

$$v(B) = \left(\frac{B}{2k\log(B)}\right)^{\frac{1}{2k\log(B)}}$$

and

$$w(B) = \left(\frac{2k\log(B)}{2\hat{d}}\right)\left(\frac{2\hat{d}}{B}\right)^{\frac{1}{2\hat{d}}}.$$

Because $v(B) \to \textit{constant}$ if $B \to \infty$ and $w(B) \to 0$ if $B \to \infty$ we have that $v(B)w(B) \to 0$ if $B \to \infty$. Hence, we can support the observation that if $B \to \infty$ then $d = \mathcal{O}(\log(B))$.

3.6 Practical aspects

To obtain the values for the optimal stepsize, one has to estimate the unknown constants σ and M_{2d} . In this section we first show that the estimated gradient is not very sensitive with respect to these constants. This means that even poor estimates of these quantities do not affect the quality of the gradient too much. We even show that the estimation is less sensitive when more evaluation points are used.

To analyze the sensitivity of the estimated gradient with respect to the unknown constants, let us assume that our estimates for σ (and M_{2d}) are $\hat{\sigma} = \kappa\sigma$ (and $\widehat{M}_{2d} = \kappa M_{2d}$), with $\kappa > 0$, respectively. Moreover, let us define the relative error $rUMSE$ as the quotient of the UMSE when the above mentioned estimates are used for one of the constants, and the UMSE when the optimal stepsize of h is used. For example

$$rUMSE_{\widehat{\sigma}}(h^*) = \frac{UMSE_{\widehat{\sigma}(h^*)}}{UMSE_{\sigma}(h^*)},$$

where the subindex $\widehat{\sigma}$ indicates that the estimated value $\widehat{\sigma}$ is used instead of σ. A similar definition holds for the other estimated constant M_{2d}. The following theorem gives expressions for these relative errors.

Theorem 3.3 *For the relative errors we have:*

$$rUMSE_{\widehat{\sigma}} = \frac{\kappa^{2-\frac{1}{d}} + (2d-1)\kappa^{-\frac{1}{d}}}{2d},$$
$$rUMSE_{\widehat{M}_{2d}} = \frac{\kappa^{-2+\frac{1}{d}} + (2d-1)\kappa^{\frac{1}{d}}}{2d}$$

Proof We first show the results when $\widehat{\sigma}$ is used. $UMSE_{\widehat{\sigma}}(h^*)$ is given in Theorem 3.1. Moreover, to calculate $UMSE(h^*)_{\sigma}$ we substitute the estimated optimal stepsize, i.e., h^* in which $\widehat{\sigma}$ is used instead of σ into the expression for the UMSE. After some tedious calculations we obtain the first part of the theorem. The second part can be obtained in a similar way. □

In Figures 3.6 and 3.7 the relative errors with respect to M_{2d} and σ, respectively, are shown for several values of d. From Figures 3.6 and 3.7 it is clear that the estimated gradients are not very sensitive with respect to the constants. For example an error of 20% for σ results into an 1.5% increase of the UMSE for $d = 2$ and an error of 50% for σ results into an 7% increase. Another important observation is that for $0 \leq \kappa \leq 2.5$ the relative errors are even decreasing in d. This means that the estimate for the derivative is more robust if we use more points for the interpolation.

To estimate the constants M_{2d} we can use the same techniques as proposed for the classical finite-difference schemes. For example in Gill *et al.* (1981)[pp. 341 - 345], an algorithm is described in which M_2 is estimated by a second-order finite-difference scheme. Of course, as described in Gill *et al.* (1981), such an estimate is not made in each iteration, since this will cost too many function evaluations, but only a few times. To estimate σ we can carry out replicates in a single point and use standard statistical methods. Again, such an estimate needs not to be made in each iteration.

3.7 Preliminary test results

For a first comparison between our method (Lagrange) and traditional central finite-differencing (CFD), we defined a set of 7 one-dimensional test functions. We are interested in estimating the derivative $f_i'(0)$. For each test function f_i we observe the function g_i, $g_i(y) = f_i(y) + \epsilon(y)$, where $\epsilon(y)$ is normally distributed with expectation $\mu = 0$ and standard deviation σ which is varied from 0.0001 to 0.1, increasing with steps of a factor 10. All results are based on 1000 replications. The test functions are listed in Table 3.4, together with their derivative in 0.

Both for Lagrange and for CFD expressions for the optimal stepsize h exist. To calculate those optimal stepsizes we would need to estimate an upper bound on the higher order derivatives (order 3 for CFD and order $2d$ for Lagrange). In these tests we

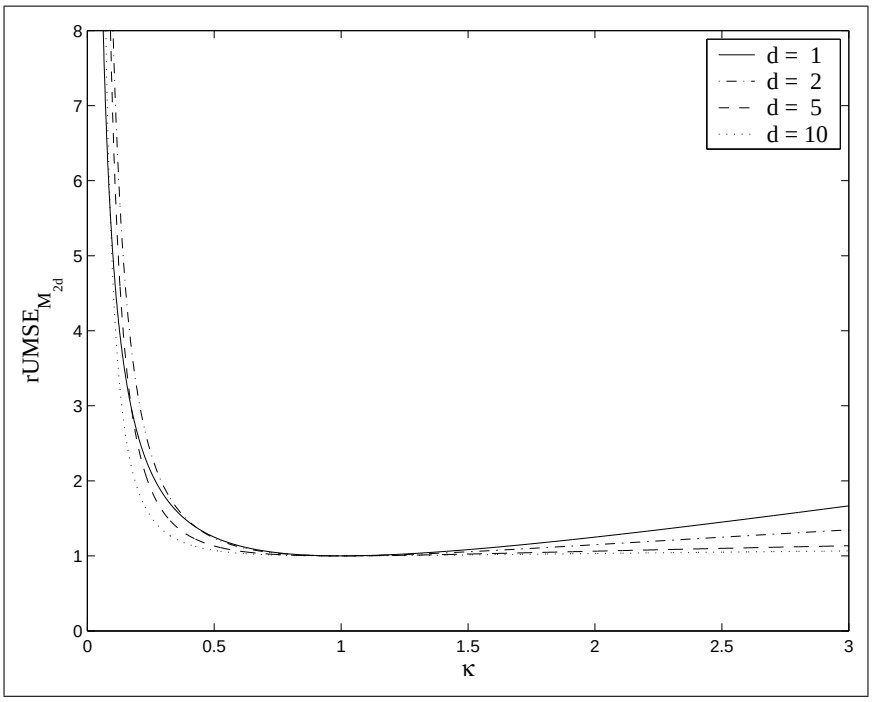

Figure 3.6: The relative errors with respect to M_{2d}.

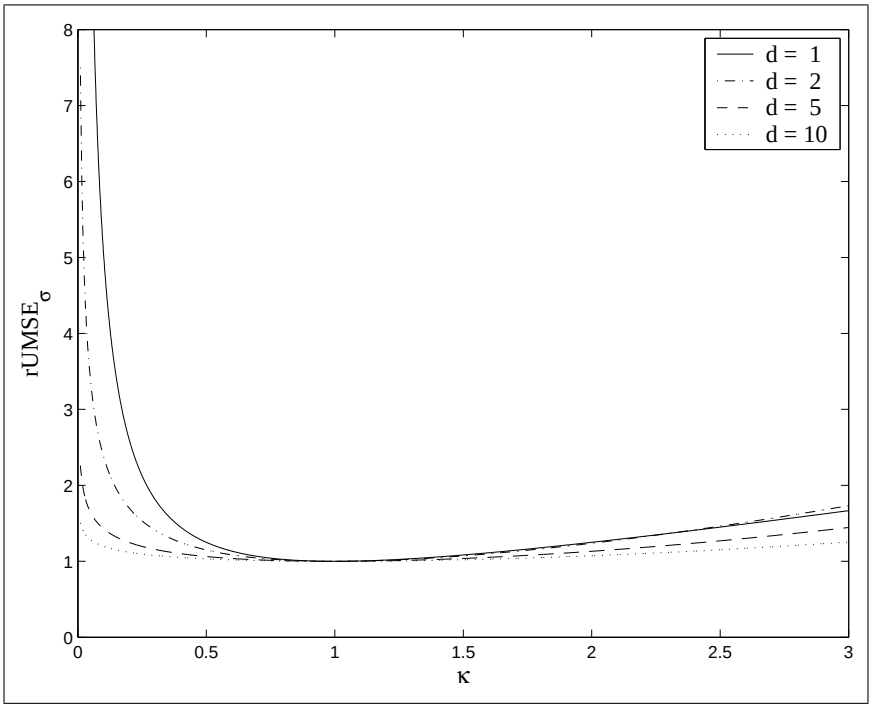

Figure 3.7: The relative errors with respect to σ.

i	f_i	$f_i'(0)$
1	$-1+e^y$	1
2	$-1+e^{3y}$	3
3	$\frac{e^y-e^{-y}}{2}$	1
4	$\cos(4(y-\frac{\pi}{8}))$	4
5	$y^4-y^3+100(1-y)^2$	-200
6	$(e^{y+1}-1)^2+(\frac{1}{\sqrt{1+(y+1)^2}}-1)^2$	$2e^2-2e-\frac{1}{2}+\frac{1}{\sqrt{2}}$
7	$\frac{\sin(24y-\frac{\pi}{8})}{12}+y$	$2\cos(\frac{-\pi}{8})+1$

Table 3.4: Test functions.

deduced the optimal stepsize for both methods by experimental grid search: we took the stepsize for which the average estimation error over 1000 experiments was smallest. For Lagrange we considered combinations of simulation budget $M = \{4, 16, 32, 128, 1024\}$ and $d = \{1, 2, 4, 8, 16\}$.

Table 3.5 shows the results for test function 1. All calculations have been carried out in double precision. The errors shown are absolute deviations from the real derivative. When errors become small, machine precision starts to play a role. Those cases have been marked with '< mp'. For $\sigma = 0$ we only included the results for $B = 32$, and set the number of replications for both methods equal to 1. As there is no noise, results for other budgets and replications are exactly the same. The results for Lagrange with $d = 1$ are exactly the same as for CFD, as the seed of the random generator has been set such that the same noise realizations occur for Lagrange and CFD. Differences for CFD between lines with the same values for σ and B are the result of different noise realizations. The error quotient for the best choice of d for the same σ and B is printed in italics. These error are most frequent larger than 1, indicating that Lagrange outperforms CFD. For higher noise levels the difference between Lagrange and CFD is smaller and the best results for Lagrange occur for low values of d. This is explained by the fact that for high noise levels the need for replication increases and the Lagrange method moves towards the CFD method by choosing a low value of d.

For the other 6 test functions, we only included the results for the best choice of d for a given B. Table 3.6 to 3.11 summarize our findings. Test function 5 draws attention, as Lagrange outperforms CFD to a much greater extent here than in the other test functions. This is not surprising because the fifth test function is polynomial. For test function 7 for $\sigma = 0.1$ Lagrange almost always chooses $d = 1$ and boils down to CFD.

3.8 Conclusions

In this research we analyzed the quality of gradient estimations obtained by using (N times replicated) Lagrange interpolation polynomials in the presence of stochastic or numerical noise. The quality of the estimations is e.g. important for nonlinear programming solvers. We took the mean-squared error as error criterion, and showed that the mean-squared error is of order $N^{-1+\frac{1}{2d}}$ if we replicate the Lagrange estimation procedure N times and use $2d$ evaluations in each replicate. As a result the order of the mean-squared error converges to N^{-1} if the number of evaluation points increases to infinity. Moreover, we showed that our approach is also useful for deterministic functions in which numerical errors are involved. We also provided an optimal division between the number of grid points and replicates in case the number of evaluations is fixed. Furthermore, we have shown that the estimation of the derivative is more robust when the number of evaluation points is increased. Finally, test results show that schemes based on Lagrange polynomials outperform central finite-differencing for low noise levels and boil down to central finite-differencing for high noise levels.

Table 3.5: Results for test function 1.

σ	M	d	h_{CFD}	h_L	E_{CFD}	E_L	E_{CFD}/E_L
0	32	1	4.01E-06	4.01E-06	3.15E-13	3.15E-13	1.00
0	32	2	4.01E-06	8.50E-04	3.15E-13	3.33E-15	94.57
0	32	4	4.01E-06	2.40E-02	3.15E-13	3.33E-16	945.67
0	32	8	4.01E-06	5.10E-02	3.15E-13	1.11E-16	**2837.00**
0	32	16	4.01E-06	5.39E-02	3.15E-13	6.66E-16	472.83
0.0001	4	1	6.00E-02	6.00E-02	8.37E-04	8.37E-04	1.00
0.0001	4	2	6.11E-02	2.79E-01	8.11E-04	3.16E-04	**2.57**
0.0001	16	1	5.11E-02	5.11E-02	5.32E-04	5.32E-04	1.00
0.0001	16	2	4.44E-02	2.26E-01	5.28E-04	1.83E-04	2.89
0.0001	16	4	4.69E-02	6.57E-01	5.23E-04	1.10E-04	4.77
0.0001	16	8	4.86E-02	1.06E+00	5.35E-04	1.04E-04	**5.14**
0.0001	32	1	4.20E-02	4.20E-02	4.04E-04	4.04E-04	1.00
0.0001	32	2	3.89E-02	2.20E-01	4.15E-04	1.33E-04	3.12
0.0001	32	4	4.07E-02	6.18E-01	4.19E-04	8.31E-05	5.04
0.0001	32	8	3.78E-02	1.07E+00	4.25E-04	7.47E-05	**5.69**
0.0001	32	16	3.94E-02	1.35E+00	4.17E-04	9.06E-05	4.61
0.0001	128	1	3.06E-02	3.06E-02	2.61E-04	2.61E-04	1.00
0.0001	128	2	2.97E-02	1.87E-01	2.71E-04	8.15E-05	3.32
0.0001	128	4	3.00E-02	5.71E-01	2.49E-04	4.39E-05	5.68
0.0001	128	8	3.11E-02	1.03E+00	2.61E-04	3.86E-05	**6.75**
0.0001	128	16	3.17E-02	1.36E+00	2.59E-04	4.40E-05	5.88
0.0001	1024	1	2.20E-02	2.20E-02	1.27E-04	1.27E-04	1.00
0.0001	1024	2	2.30E-02	1.51E-01	1.34E-04	3.47E-05	3.85
0.0001	1024	4	2.30E-02	4.86E-01	1.31E-04	1.80E-05	7.31

Continued on next page

Table 3.5 – continued from previous page

σ	M	d	h_{CFD}	h_L	E_{CFD}	E_L	E_{CFD}/E_L
0.0001	1024	8	2.18E-02	9.71E-01	1.32E-04	1.49E-05	**8.89**
0.0001	1024	16	2.20E-02	1.31E+00	1.29E-04	1.61E-05	8.02
0.001	4	1	1.29E-01	1.29E-01	3.82E-03	3.82E-03	1.00
0.001	4	2	1.26E-01	4.34E-01	3.85E-03	2.00E-03	**1.93**
0.001	16	1	1.03E-01	1.03E-01	2.47E-03	2.47E-03	1.00
0.001	16	2	9.44E-02	3.86E-01	2.39E-03	1.16E-03	2.06
0.001	16	4	1.01E-01	8.30E-01	2.35E-03	8.41E-04	2.79
0.001	16	8	1.03E-01	1.22E+00	2.47E-03	8.66E-04	**2.85**
0.001	32	1	9.33E-02	9.33E-02	1.94E-03	1.94E-03	1.00
0.001	32	2	9.00E-02	3.47E-01	1.87E-03	8.87E-04	2.11
0.001	32	4	8.93E-02	7.61E-01	1.91E-03	6.23E-04	**3.06**
0.001	32	8	8.67E-02	1.16E+00	1.91E-03	6.24E-04	3.05
0.001	32	16	8.70E-02	1.39E+00	1.90E-03	8.54E-04	2.22
0.001	128	1	7.41E-02	7.41E-02	1.20E-03	1.20E-03	1.00
0.001	128	2	6.99E-02	2.99E-01	1.21E-03	5.04E-04	2.40
0.001	128	4	6.99E-02	6.99E-01	1.20E-03	3.49E-04	3.45
0.001	128	8	6.89E-02	1.16E+00	1.19E-03	3.43E-04	**3.48**
0.001	128	16	8.00E-02	1.42E+00	1.26E-03	4.26E-04	2.95
0.001	1024	1	5.01E-02	5.01E-02	6.29E-04	6.29E-04	1.00
0.001	1024	2	4.83E-02	2.42E-01	6.19E-04	2.15E-04	2.88
0.001	1024	4	5.33E-02	6.63E-01	6.51E-04	1.37E-04	**4.77**
0.001	1024	8	5.16E-02	1.06E+00	6.04E-04	1.28E-04	4.70
0.001	1024	16	5.13E-02	1.41E+00	6.23E-04	1.49E-04	4.17
0.01	4	1	2.72E-01	2.72E-01	1.76E-02	1.76E-02	1.00
0.01	4	2	2.78E-01	6.38E-01	1.85E-02	1.29E-02	**1.43**
0.01	16	1	2.16E-01	2.16E-01	1.14E-02	1.14E-02	1.00
0.01	16	2	2.12E-01	5.67E-01	1.09E-02	7.05E-03	1.54
0.01	16	4	2.06E-01	1.03E+00	1.16E-02	6.82E-03	**1.70**
0.01	16	8	2.08E-01	1.37E+00	1.12E-02	8.00E-03	1.40
0.01	32	1	2.00E-01	2.00E-01	9.04E-03	9.04E-03	1.00
0.01	32	2	2.06E-01	5.42E-01	9.06E-03	5.59E-03	1.62
0.01	32	4	2.01E-01	1.02E+00	8.70E-03	5.01E-03	**1.74**
0.01	32	8	1.91E-01	1.35E+00	8.90E-03	5.73E-03	1.55
0.01	32	16	1.89E-01	1.56E+00	8.67E-03	7.60E-03	1.14
0.01	128	1	1.66E-01	1.66E-01	5.72E-03	5.72E-03	1.00
0.01	128	2	1.53E-01	4.94E-01	5.63E-03	3.14E-03	1.79
0.01	128	4	1.66E-01	9.46E-01	5.51E-03	2.67E-03	**2.06**
0.01	128	8	1.47E-01	1.31E+00	5.76E-03	2.97E-03	1.94
0.01	128	16	1.66E-01	1.52E+00	5.69E-03	3.86E-03	1.48
0.01	1024	1	1.17E-01	1.17E-01	2.86E-03	2.86E-03	1.00
0.01	1024	2	1.07E-01	3.91E-01	2.87E-03	1.37E-03	2.09
0.01	1024	4	1.02E-01	8.28E-01	2.74E-03	1.01E-03	**2.71**
0.01	1024	8	1.06E-01	1.20E+00	2.85E-03	1.12E-03	2.55
0.01	1024	16	1.08E-01	1.46E+00	2.84E-03	1.42E-03	2.00
0.1	4	1	5.78E-01	5.78E-01	8.55E-02	8.55E-02	1.00

Continued on next page

Table 3.5 – continued from previous page

σ	M	d	h_{CFD}	h_L	E_{CFD}	E_L	E_{CFD}/E_L
0.1	4	2	5.66E-01	9.87E-01	8.47E-02	8.46E-02	**1.00**
0.1	16	1	4.69E-01	4.69E-01	5.18E-02	5.18E-02	1.00
0.1	16	2	4.46E-01	8.84E-01	5.48E-02	4.68E-02	**1.17**
0.1	16	4	4.50E-01	1.24E+00	5.39E-02	5.67E-02	0.95
0.1	16	8	4.42E-01	1.54E+00	5.18E-02	6.97E-02	0.74
0.1	32	1	3.96E-01	3.96E-01	4.00E-02	4.00E-02	1.00
0.1	32	2	3.91E-01	8.91E-01	4.22E-02	3.61E-02	**1.17**
0.1	32	4	3.68E-01	1.18E+00	4.42E-02	4.21E-02	1.05
0.1	32	8	4.24E-01	1.52E+00	4.05E-02	5.01E-02	0.81
0.1	32	16	3.71E-01	1.66E+00	4.13E-02	7.01E-02	0.59
0.1	128	1	3.50E-01	3.50E-01	2.65E-02	2.65E-02	1.00
0.1	128	2	3.12E-01	7.11E-01	2.63E-02	2.05E-02	1.28
0.1	128	4	3.21E-01	1.16E+00	2.71E-02	2.10E-02	**1.29**
0.1	128	8	3.58E-01	1.44E+00	2.65E-02	2.59E-02	1.02
0.1	128	16	3.51E-01	1.63E+00	2.57E-02	3.62E-02	0.71
0.1	1024	1	2.33E-01	2.33E-01	1.33E-02	1.33E-02	1.00
0.1	1024	2	2.41E-01	6.30E-01	1.30E-02	8.45E-03	1.54
0.1	1024	4	2.27E-01	1.05E+00	1.27E-02	7.90E-03	**1.60**
0.1	1024	8	2.43E-01	1.41E+00	1.31E-02	9.69E-03	1.35
0.1	1024	16	2.37E-01	1.58E+00	1.30E-02	1.31E-02	0.99

σ	M	d	h_{CFD}	h_L	E_{CFD}	E_L	E_{CFD}/E_L
0	4	2	1.44E-06	2.14E-04	1.59E-12	8.88E-16	1785.00
0	16	8	1.44E-06	3.09E-02	1.58E-12	$<$ mp	-
0	32	8	1.44E-06	3.09E-02	1.58E-12	$<$ mp	-
0	128	8	1.44E-06	3.09E-02	1.59E-12	$<$ mp	-
0	1024	8	1.44E-06	3.09E-02	1.59E-12	$<$ mp	-
0.0001	4	2	1.79E-02	9.30E-02	2.49E-03	9.37E-04	2.65
0.0001	16	8	1.67E-02	3.51E-01	1.58E-03	2.98E-04	5.31
0.0001	32	8	1.38E-02	3.37E-01	1.24E-03	2.25E-04	5.51
0.0001	128	8	1.16E-02	3.43E-01	8.13E-04	1.19E-04	6.83
0.0001	1024	8	8.00E-03	3.22E-01	3.97E-04	4.42E-05	9.00
0.001	4	2	4.08E-02	1.32E-01	1.17E-02	6.11E-03	1.92
0.001	16	4	3.44E-02	2.66E-01	7.31E-03	2.61E-03	2.80
0.001	32	4	2.76E-02	2.64E-01	5.63E-03	1.84E-03	3.06
0.001	128	4	2.56E-02	2.44E-01	3.76E-03	1.02E-03	3.68
0.001	1024	8	1.78E-02	3.57E-01	1.88E-03	3.83E-04	4.90
0.01	4	2	8.68E-02	2.26E-01	5.30E-02	3.79E-02	1.40
0.01	16	4	7.59E-02	3.39E-01	3.49E-02	2.12E-02	1.65
0.01	32	4	6.59E-02	3.40E-01	2.64E-02	1.54E-02	1.71
0.01	128	4	4.91E-02	3.14E-01	1.69E-02	7.97E-03	2.12
0.01	1024	4	3.67E-02	2.86E-01	8.57E-03	3.17E-03	2.70
0.1	4	2	2.01E-01	3.19E-01	2.46E-01	2.42E-01	1.02
0.1	16	2	1.70E-01	3.00E-01	1.65E-01	1.43E-01	1.16
0.1	32	2	1.35E-01	2.80E-01	1.28E-01	1.06E-01	1.21
0.1	128	2	1.11E-01	2.51E-01	7.90E-02	6.04E-02	1.31
0.1	1024	4	7.33E-02	3.50E-01	3.98E-02	2.43E-02	1.64

Table 3.6: Results for test function 2.

σ	M	d	h_{CFD}	h_L	E_{CFD}	E_L	E_{CFD}/E_L
0	4	2	4.01E-06	8.50E-04	3.15E-13	3.33E-15	94.57
0	16	4	4.01E-06	2.40E-02	3.15E-13	$<$ mp	-
0	32	4	4.01E-06	2.40E-02	3.15E-13	$<$ mp	-
0	128	4	4.01E-06	2.40E-02	3.15E-13	$<$ mp	-
0	1024	4	4.01E-06	2.40E-02	3.15E-13	$<$ mp	-
0.0001	4	2	6.00E-02	2.74E-01	8.22E-04	3.14E-04	2.62
0.0001	16	8	4.67E-02	1.06E+00	5.23E-04	1.02E-04	5.11
0.0001	32	8	4.49E-02	1.02E+00	4.20E-04	7.66E-05	5.48
0.0001	128	8	3.02E-02	9.68E-01	2.67E-04	4.09E-05	6.54
0.0001	1024	8	2.43E-02	9.64E-01	1.31E-04	1.39E-05	9.48
0.001	4	2	1.31E-01	4.30E-01	3.91E-03	1.99E-03	1.97
0.001	16	4	1.01E-01	7.78E-01	2.43E-03	8.24E-04	2.95
0.001	32	8	9.33E-02	1.21E+00	1.94E-03	6.34E-04	3.06
0.001	128	8	7.78E-02	1.16E+00	1.24E-03	3.40E-04	3.65
0.001	1024	8	5.11E-02	1.11E+00	6.08E-04	1.32E-04	4.61
0.01	4	2	2.66E-01	6.56E-01	1.85E-02	1.32E-02	1.39
0.01	16	4	2.36E-01	1.03E+00	1.17E-02	6.61E-03	1.77
0.01	32	4	1.84E-01	1.01E+00	8.87E-03	5.00E-03	1.77
0.01	128	4	1.44E-01	9.48E-01	5.76E-03	2.72E-03	2.12
0.01	1024	4	1.04E-01	8.24E-01	2.77E-03	9.97E-04	2.78
0.1	4	2	5.87E-01	9.92E-01	8.40E-02	8.04E-02	1.04
0.1	16	2	4.70E-01	9.46E-01	5.17E-02	4.72E-02	1.10
0.1	32	2	4.02E-01	9.01E-01	4.10E-02	3.53E-02	1.16
0.1	128	2	3.24E-01	7.39E-01	2.69E-02	2.00E-02	1.34
0.1	1024	4	2.36E-01	1.03E+00	1.29E-02	8.33E-03	1.55

Table 3.7: Results for test function 3.

σ	M	d	h_{CFD}	h_L	E_{CFD}	E_L	E_{CFD}/E_L
0	4	2	2.17E-06	2.34E-04	6.83E-12	1.15E-14	591.81
0	16	4	2.17E-06	5.50E-03	6.83E-12	4.44E-16	15387.00
0	32	4	2.17E-06	5.50E-03	6.83E-12	4.44E-16	15387.00
0	128	4	2.17E-06	5.50E-03	6.83E-12	4.44E-16	15387.00
0	1024	4	2.17E-06	5.50E-03	6.83E-12	4.44E-16	15387.00
0.0001	4	2	1.31E-02	6.69E-02	3.44E-03	1.26E-03	2.74
0.0001	16	8	1.22E-02	3.00E-01	2.07E-03	3.93E-04	5.26
0.0001	32	8	1.08E-02	2.86E-01	1.70E-03	2.65E-04	6.42
0.0001	128	16	8.78E-03	4.09E-01	1.07E-03	1.48E-04	7.23
0.0001	1024	16	5.39E-03	3.95E-01	5.47E-04	5.32E-05	10.28
0.001	4	2	3.20E-02	1.08E-01	1.58E-02	8.11E-03	1.94
0.001	16	8	2.53E-02	3.42E-01	9.50E-03	3.13E-03	3.03
0.001	32	8	2.23E-02	3.28E-01	7.68E-03	2.40E-03	3.19
0.001	128	4	1.88E-02	1.89E-01	4.99E-03	1.33E-03	3.76
0.001	1024	8	1.11E-02	2.93E-01	2.53E-03	4.62E-04	5.47
0.01	4	2	6.36E-02	1.69E-01	7.30E-02	5.01E-02	1.46
0.01	16	4	4.78E-02	2.82E-01	4.62E-02	2.51E-02	1.84
0.01	32	4	4.78E-02	2.57E-01	3.56E-02	1.88E-02	1.90
0.01	128	4	3.90E-02	2.38E-01	2.29E-02	9.97E-03	2.29
0.01	1024	8	2.71E-02	3.41E-01	1.16E-02	3.91E-03	2.96
0.1	4	2	1.56E-01	2.78E-01	3.37E-01	3.13E-01	1.08
0.1	16	4	1.27E-01	3.73E-01	2.10E-01	1.80E-01	1.17
0.1	32	4	1.04E-01	3.49E-01	1.71E-01	1.42E-01	1.21
0.1	128	2	7.99E-02	1.79E-01	1.07E-01	8.06E-02	1.33
0.1	1024	4	5.40E-02	2.87E-01	5.40E-02	3.08E-02	1.75

Table 3.8: Results for test function 4.

σ	M	d	h_{CFD}	h_L	E_{CFD}	E_L	E_{CFD}/E_L
0	4	2	1.81E-05	1.51E-01	7.67E-13	2.84E-14	27.00
0	16	2	1.81E-05	1.51E-01	7.67E-13	2.84E-14	27.00
0	32	16	1.81E-05	2.80E-01	7.67E-13	$<$ mp	-
0	128	16	1.81E-05	2.80E-01	7.67E-13	$<$ mp	-
0	1024	16	1.81E-05	2.80E-01	7.67E-13	$<$ mp	-
0.0001	4	2	3.18E-02	1.76E+00	1.53E-03	4.38E-05	35.04
0.0001	16	2	2.56E-02	1.77E+00	9.46E-04	2.15E-05	43.94
0.0001	32	2	2.29E-02	1.81E+00	7.18E-04	1.44E-05	49.80
0.0001	128	2	1.74E-02	1.80E+00	4.73E-04	7.35E-06	64.40
0.0001	1024	2	1.37E-02	1.78E+00	2.39E-04	2.75E-06	86.77
0.001	4	2	6.89E-02	1.78E+00	6.85E-03	3.96E-04	17.29
0.001	16	2	5.97E-02	1.79E+00	4.41E-03	2.16E-04	20.46
0.001	32	2	4.74E-02	1.80E+00	3.62E-03	1.54E-04	23.43
0.001	128	2	3.89E-02	1.80E+00	2.26E-03	7.63E-05	29.62
0.001	1024	2	2.89E-02	1.80E+00	1.13E-03	2.64E-05	42.74
0.01	4	2	1.44E-01	1.76E+00	3.22E-02	4.41E-03	7.30
0.01	16	2	1.16E-01	1.80E+00	1.97E-02	2.05E-03	9.59
0.01	32	2	9.91E-02	1.78E+00	1.62E-02	1.46E-03	11.04
0.01	128	2	8.53E-02	1.78E+00	1.02E-02	7.60E-04	13.47
0.01	1024	2	5.90E-02	1.80E+00	5.06E-03	2.65E-04	19.09
0.1	4	2	3.38E-01	1.76E+00	1.48E-01	4.36E-02	3.39
0.1	16	2	2.66E-01	1.78E+00	9.53E-02	2.17E-02	4.39
0.1	32	2	2.07E-01	1.76E+00	7.32E-02	1.53E-02	4.79
0.1	128	2	1.87E-01	1.77E+00	4.60E-02	7.51E-03	6.12
0.1	1024	2	1.24E-01	1.79E+00	2.34E-02	2.70E-03	8.65

Table 3.9: Results for test function 5.

σ	M	d	h_{CFD}	h_L	E_{CFD}	E_L	E_{CFD}/E_L
0	4	2	3.48E-06	5.90E-04	1.29E-11	6.39E-14	202.06
0	16	8	3.48E-06	4.65E-02	1.29E-11	1.78E-15	7274.00
0	32	8	3.48E-06	4.65E-02	1.29E-11	1.78E-15	7274.00
0	128	8	3.48E-06	4.65E-02	1.29E-11	1.78E-15	7274.00
0	1024	8	3.48E-06	4.65E-02	1.29E-11	1.78E-15	7274.00
0.0001	4	2	1.52E-02	8.74E-02	3.13E-03	9.57E-04	3.27
0.0001	16	8	1.14E-02	5.03E-01	2.02E-03	2.11E-04	9.58
0.0001	32	8	1.16E-02	5.04E-01	1.58E-03	1.49E-04	10.57
0.0001	128	4	9.12E-03	2.64E-01	9.60E-04	9.23E-05	10.41
0.0001	1024	4	5.76E-03	2.49E-01	4.95E-04	3.23E-05	15.34
0.001	4	2	3.41E-02	1.38E-01	1.45E-02	5.94E-03	2.44
0.001	16	4	2.73E-02	3.21E-01	9.15E-03	1.97E-03	4.65
0.001	32	8	2.38E-02	5.03E-01	7.54E-03	1.49E-03	5.06
0.001	128	4	1.94E-02	3.12E-01	4.71E-03	7.96E-04	5.92
0.001	1024	8	1.21E-02	5.02E-01	2.34E-03	2.71E-04	8.62
0.01	4	2	7.90E-02	2.40E-01	6.95E-02	3.77E-02	1.84
0.01	16	4	5.88E-02	4.16E-01	4.36E-02	1.67E-02	2.61
0.01	32	4	4.81E-02	3.97E-01	3.33E-02	1.25E-02	2.68
0.01	128	4	4.17E-02	3.74E-01	2.11E-02	6.19E-03	3.41
0.01	1024	4	2.63E-02	3.44E-01	1.06E-02	2.52E-03	4.23
0.1	4	2	1.60E-01	3.61E-01	3.17E-01	2.42E-01	1.31
0.1	16	4	1.20E-01	5.12E-01	1.93E-01	1.30E-01	1.48
0.1	32	4	1.09E-01	4.81E-01	1.52E-01	9.41E-02	1.62
0.1	128	4	7.78E-02	4.93E-01	9.99E-02	5.44E-02	1.84
0.1	1024	4	5.56E-02	4.16E-01	4.97E-02	2.04E-02	2.44

Table 3.10: Results for test function 6.

σ	M	d	h_{CFD}	h_L	E_{CFD}	E_L	E_{CFD}/E_L
0	4	2	1.00E-08	4.71E-05	3.13E-12	1.38E-14	227.65
0	16	8	1.00E-08	8.00E-03	3.13E-12	1.78E-15	1764.25
0	32	16	1.00E-08	1.81E-02	3.13E-12	1.33E-15	2352.33
0	128	16	1.00E-08	1.81E-02	3.13E-12	1.33E-15	2352.33
0	1024	16	1.00E-08	1.81E-02	3.13E-12	1.33E-15	2352.33
0.0001	4	2	6.11E-03	1.86E-02	8.36E-03	4.39E-03	1.90
0.0001	16	8	4.67E-03	5.78E-02	5.16E-03	1.85E-03	2.79
0.0001	32	8	4.11E-03	5.46E-02	4.37E-03	1.38E-03	3.17
0.0001	128	8	3.44E-03	5.37E-02	2.77E-03	7.31E-04	3.79
0.0001	1024	16	2.36E-03	6.94E-02	1.35E-03	3.00E-04	4.50
0.001	4	2	1.26E-02	3.02E-02	3.81E-02	2.80E-02	1.36
0.001	16	4	1.02E-02	4.80E-02	2.49E-02	1.41E-02	1.76
0.001	32	4	8.56E-03	4.43E-02	2.00E-02	1.07E-02	1.87
0.001	128	4	6.78E-03	4.06E-02	1.27E-02	6.03E-03	2.11
0.001	1024	8	5.04E-03	5.97E-02	6.45E-03	2.34E-03	2.75
0.01	4	2	2.61E-02	4.89E-02	1.78E-01	1.68E-01	1.06
0.01	16	2	2.08E-02	4.33E-02	1.15E-01	1.02E-01	1.13
0.01	32	2	1.83E-02	3.91E-02	8.84E-02	7.37E-02	1.20
0.01	128	2	1.44E-02	3.24E-02	5.69E-02	4.37E-02	1.30
0.01	1024	4	1.12E-02	4.66E-02	2.94E-02	1.91E-02	1.55
0.1	4	1	6.33E-02	6.33E-02	8.42E-01	8.42E-01	1.00
0.1	16	1	4.56E-02	4.56E-02	5.19E-01	5.19E-01	1.00
0.1	32	1	4.12E-02	4.12E-02	4.05E-01	4.05E-01	1.00
0.1	128	1	3.17E-02	3.17E-02	2.75E-01	2.75E-01	1.00
0.1	1024	2	2.37E-02	4.44E-02	1.30E-01	1.21E-01	1.07

Table 3.11: Results for test function 7.

Chapter 4

Constrained optimization involving expensive function evaluations: a sequential approach

Abstract: This paper presents a new sequential method for constrained nonlinear optimization problems. The principal characteristics of these problems are very time consuming function evaluations and the absence of derivative information. Such problems are common in design optimization, where time consuming function evaluations are carried out by simulation tools (e.g., FEM, CFD). Classical optimization methods, based on derivatives, are not applicable because often derivative information is not available and is too expensive to approximate through finite-differencing.

The algorithm first creates an experimental design. In the design points the underlying functions are evaluated. Local linear approximations of the real model are obtained with help of weighted regression techniques. The approximating model is then optimized within a trust region to find the best feasible objective improving point. This trust region moves along the most promising direction, which is determined on the basis of the evaluated objective values and constraint violations combined in a filter criterion. If the geometry of the points that determine the local approximations becomes bad, i.e. the points are located in such a way that they result in a bad approximation of the actual model, then we evaluate a geometry improving instead of an objective improving point. In each iteration a new local linear approximation is built, and either a new point is evaluated (objective or geometry improving) or the trust region is decreased. Convergence of the algorithm is guided by the size of this trust region. The focus of the approach is on getting good solutions with a limited number of function evaluations.

4.1 Introduction

In the past, design merely consisted of experimentation and physical prototyping. In the last decade, physical computer simulations, such as finite element methods, are widely used in engineering design and analysis. It is impressive to see the enormous market for these Computer Aided Engineering (CAE) tools. Examples of CAE tools can be found in Aerodynamics, Computational Fluid Dynamics, Computational Electromagnetics, Mechanical Engineering, and Electronic Circuit Simulations.

CAE tools enable the designers to simulate the performance of a product or process, but still designers are confronted with the problem of finding settings for a, possibly large, number of design parameters. These parameters should be set optimal with respect to

several simulated product or process characteristics. These characteristics, called response parameters , may originate from different engineering disciplines. Since there are many possible design parameter settings and computer simulations are time consuming in many cases, the crucial question becomes how to find the best possible setting with a minimum number of simulations.

In this paper we consider such constrained optimization problems that contain non-linear functions that require expensive function evaluations (i.e. simulations). The characteristic feature of the problems we consider is that in many cases simulations do not yield any derivative information. Moreover, there is no explicit information about the functional relationship between the designs and the responses simulated by the black-box computer model. Finally, we only consider optimization problems arising from deterministic simulations. In Brekelmans *et al.* (2001) the authors introduced a new optimization method for this type of problems. This paper describes the algorithm in greater detail and presents many sophistications and extensions to the algorithm.

Classical optimization methods, based on derivatives, are not applicable because often derivative information is not available and is too expensive to approximate through finite-differencing. Hence, the optimization algorithm has to be derivative free. For an overview of derivative free methods see Lewis *et al.* (2001). Most of the recent derivative free trust-region methods are not applicable to the constrained optimization problem that we are interested in. Powell (2002), Conn *et al.* (1997), Dennis and Torczon (1994), Booker *et al.* (1999) and Torczon (1992) focus on unconstrained optimization or on constrained problems for which the constraints are explicitly known. In Audet *et al.* (2000) the pattern search method (Torczon (1992)) is extended for problems with black-box constraints by using a filter approach. We consider problems in which there are also constraints on the responses simulated by the black-box computer model. Additionally, the quadratic models which some of these authors advocate, require exactly $\frac{1}{2}(n+1)(n+2)$ observations, where n is the dimension of the design space. For many black-box optimization problems, especially with a large number of design variables or significant time/cost of black-box evaluations, this number is too large for practical applications.

We compare our method with Powell's COBYLA (Powell (1994a)), since this method is also applicable to the problems considered in this paper. His method only requires $n+1$ observations and can also handle constraints on the simulated responses. Very recently in Powell (2003) and Powell (2004) a promising new method for unconstrained problems is proposed, which builds up a quadratic approximation of the objective by using a sort of quasi-Newton updating approach.

Algorithms designed for global optimization (e.g., Jones *et al.* (1998)) may require many function evaluations in the entire feasible region if the problem has many variables or many local optima. See also Jones (2001b) for a good review on such methods which use

global response surface approximations. In our view such methods are likely to be quite effective for low-dimensional problems (say up to 20 variables), but suffer from a curse of dimensionality that may make them less effective for larger problems. Therefore, for practical reasons, we have to restrict ourselves to finding local optima of the optimization problem. Moreover, we will not aim at high accuracy. Since high accuracy requires many function evaluations in the neighborhood of an optimum, which is in our case too expensive. Besides that, in many cases high accuracy is not attained due to the inaccuracy of the simulation model and in other cases it is even not necessary to be very accurate because of the practical applications the problem is based on. In Toropov's multi-point approximation method (see Toropov *et al.* (1993)) in each iteration a series of n new simulations is performed to obtain a new local approximation. See also Etman (1997). Since for the problems we consider each simulation is time consuming, also this method requires too many simulations. In our method simulations to improve the local approximation are performed adaptively. This means that a simulation to improve the approximation is performed only when the current geometry of the design points simulated already is too bad.

Besides iterative algorithms which are often based on local approximations, also non-iterative algorithms based on global approximations are proposed in the literature. See e.g. Schoofs *et al.* (1994), Balabanov *et al.* (1999), and Den Hertog and Stehouwer (2002a). In these methods explicit global approximations for the functional relationships between the designs and the responses are made. By substituting these relationships into the original design optimization problem, an explicit nonlinear optimization remains to be solved. Key issues in these approaches are the choice of the points which should be simulated to get maximal global information and which kinds of global approximation models to use. The advantages of global approximation methods is that global insight is obtained on the behavior of responses. Moreover, when the design optimization problem changes somewhat (e.g. a change in a bound on a response), no new simulations have to be carried out since the global approximations can be used again. The disadvantage of this approach, however, is the high number of simulations required to obtain good global approximations. We therefore focus on an iterative method which uses local approximations.

Other approaches to the design optimization problem, which are used especially in the field of discrete-event simulation, are for example local search, tabu search, simulated annealing (e.g., Glover *et al.* (1996)). These methods, however, assume that one simulation run can be performed quickly and often only deal with (simple) constraints on the design parameters. For the problems we are interested in, i.e. design problems with time consuming simulation runs, these methods require too many runs. Moreover, we will consider the constrained case.

This paper presents an iterative optimization algorithm that fits into the general trust-

region framework (see Conn *et al.* (2000)). Moreover, we report on the preliminary results of our toolbox SEQUEM, in which this new algorithm has been implemented. Some of the key elements of the algorithm are inspired by the filter method (see Fletcher and Leyffer (2002)) which jointly considers the objective value and constraints of the optimization problem. The filter method is used for the selection of the current iterate, and the computation and selection of new designs for the next function evaluation.

This paper is organized as follows. In Section 4.2 the mathematical formulation of the design optimization problem is given. Section 4.3 describes the steps of the algorithm. Numerical results are presented in Section 4.4. Section 4.5 concludes.

4.2 Problem description

In this paper the problem of black-box optimization is considered. The main characteristic of a black-box optimization problem is that the objective and constraint functions are implicit functions of the design and response variables. The functional relation between the designs and the responses is not explicitly known. This situation often occurs when dealing with simulation models. A set of design parameters $x \in X \subseteq \mathbb{R}^n$ is fed to the black-box, for example a simulation tool, and a set of response parameters $r(x) \in \mathbb{R}^m$ is returned according to some unknown relationship between the designs and responses. The set X is the domain on which the black-box machine returns sensible responses. A design vector $x \notin X$ is not guaranteed to produce sensible output, or may produce no output at all. The black-box process is illustrated by Figure 4.1. In many applications the black-box process is very time consuming or costly. Furthermore, there is no derivative information available for the responses $r(x)$.

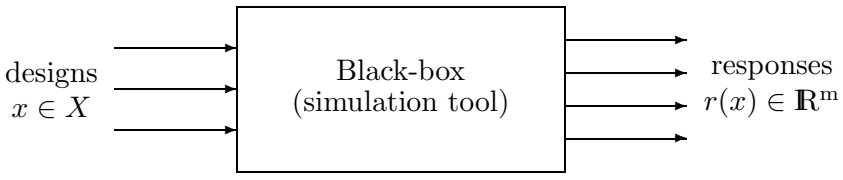

Figure 4.1: Black-box optimization.

Goal of the optimization problem is to minimize (maximize) a given function of the design and response parameters, subject to a set of constraints regarding (combinations of) the design and response parameters. The black-box minimization problem can be

mathematically formulated as follows:

$$\min_{x \in X} z = f_0(x, r(x)) \tag{4.1a}$$

$$\text{s.t. } l_i^f \leq f_i(x, r(x)) \leq u_i^f, \qquad 1 \leq i \leq m_f, \tag{4.1b}$$

$$l_j^g \leq g_j(x) \leq u_j^g, \qquad 1 \leq j \leq m_g, \tag{4.1c}$$

$$l^x \leq x \leq u^x. \tag{4.1d}$$

In this model it is assumed that the functions f_i $(0 \leq i \leq m_f)$ and g_j $(1 \leq j \leq m_g)$ are known. Note that this does not imply that problem (4.1) is explicitly known, since it still contains the unknown function $r(x)$ coming from the black-box machine. Moreover, it is not required that the lower and upper bounds in (4.1b) and (4.1c) are finite. The bounds l^x and u^x, however, are assumed to be finite. The feasible region of the design constraints, (4.1c) and (4.1d), is assumed to completely lie within X. For convenience it is assumed that X is defined by the feasible region of constraints (4.1c) and (4.1d).

4.3 Sequential algorithm

In this section we describe the sequential algorithm SEQUEM, which is a variant of the trust-region method. See Conn *et al.* (2000) for an extensive discussion of trust-region methods. The main steps of the SEQUEM algorithm are shown in Algorithm 1 below:

Algorithm 1 The main steps of the SEQUEM algorithm

Initialization (See Section 4.3.2)
Call black-box (n+1 times)
repeat
  Select current iterate (See Section 4.3.3)
  Approximate (local) model (See Section 4.3.4)
  Compute filter improving designs (See Section 4.3.5)
  Compute geometry improving designs (See Section 4.3.6)
  if Selected design good **then** (See Section 4.3.7)
    Call black-box (1 time)
  else
    Decrease trust region (See Section 4.3.7)
  end if
until stop criterion satisfied
end

A new aspect, compared to the standard trust-region algorithm, is the use of the filter method for both the selection of the current iterate and for the trial step. The filter method introduced by Fletcher and Leyffer (2002) enables the optimization of a constrained problem without the use of a penalty function.

In the remainder of this section the steps of the algorithm are described in greater detail. It should be noted that most steps of the algorithm are in a great extend independent of each other. Therefore, it is possible to use different implementations for these steps. The SEQUEM optimization package contains many different implementations of the steps in the basic trust-region algorithm.

4.3.1 Preparations

Besides problem definition (4.1) the optimization method discussed in this paper requires additional information about the problem, which is important for finding an optimum in an efficient manner. The additional information assumed to be available is as follows:

Start design: $x^{(0)} \in X$. In practice, designers often have a good idea about, or already work with a design that yields nice results, but which has not been optimized yet. Therefore, this design provides the starting point for the algorithm.

Initial step size: $\Delta^{(0)} \in \mathbb{R}^n$. Initial step sizes are needed to choose the step sizes taken in each dimension. The initial step sizes should be such that the effect on the responses will be of comparable magnitude. Because in the algorithm the trust region is decreased with the same factor in all dimensions, the step sizes can be seen as a way of scaling the design parameters.

Constraint scaling: One of the ingredients of the algorithm is that constraint violations are allowed during the execution of the algorithm. This can shorten the path to the solution of the problem. To be able to exploit constraint violations to a maximum extent it is important that different constraint violations can be compared using a certain measure. For ease of notation it is assumed that the constraints in (4.1b) are already scaled in such a way that a constraint violation of magnitude 1 for $f_i(x, r(x))$ and for $f_j(x, r(x))$ $(i \neq j)$ are of similar relevance.

4.3.2 Initialization

Before the main body of the algorithm can be entered several actions have to be performed. Firstly, a collection of design vectors has to be evaluated by the black-box machine to be able to approximate $r(x)$. Hereto, we use the start design $x^{(0)}$ and the initial step sizes $\Delta^{(0)}$. We initialize the search with $n + 1$ vectors, consisting of the starting point

$x^{(0)}$ and n additional vectors $x^{(1)}, ..., x^{(n)}$. The coordinates of $x^{(i)}, i = 1, ..., n$, are given by $x_j^{(i)} = x_j^{(0)} + a_{ij}\Delta_j^{(0)}$, where a_{ij} is either -1 or +1. The $n \times n$ matrix of numbers a_{ij} is computed by finding a two level (-1 or +1) D-optimal design based on a first order polynomial model.

Secondly, an initial *trust region* has to be defined. Throughout the algorithm a box-shaped trust region is used, which is completely determined by its center $x^{(c)} \in \mathbb{R}^n$ and step sizes $\Delta \in \mathbb{R}^n$. The box-shaped trust region is defined by

$$TR(x^{(c)}, \Delta) := \left\{x \in \mathbb{R}^n : |x_j - x_j^{(c)}| \leq \Delta_j,\ 1 \leq j \leq n\right\}.$$

It seems natural to let the size of the initial trust region depend on the size of $\Delta^{(0)}$. Also we would like to use our estimates in a slightly larger region than the region in which we have performed the initial evaluations. Therefore, the initial trust region after k black-box evaluations is chosen as $\Delta^{(k)} = \delta\Delta^{(0)}$ with $\delta \geq 1$. Note that the center of the trust region is determined by the current iterate which is selected in the first step of the main algorithm (see Section 4.3.3).

4.3.3 Selection of current iterate (filter method)

The current iterate is a very important element of the algorithm as it determines the center of the trust region in which our next black-box evaluation will take place. At the termination of the algorithm, the sequence of selected current iterates forms a path from the start design to the final approximation of the optimum of problem (4.1). The idea is to select that design from the set of evaluated designs as current iterate that is closest to the optimum of problem (4.1). However, in practice the exact location of the optimum is not known, and consequently it is impossible to determine which of the evaluated designs is closest to it. Therefore, the algorithm has to select the most promising design based upon alternative criteria.

If a new black-box evaluation has been carried out, then the corresponding design is not guaranteed to be the current iterate for the next iteration. A new evaluation might result in a worse objective value than the objective value at the current iterate, for example because of a bad approximation of the local model. In such a case it may be wise not to shift the center of the trust region to this new design but stick to the old current iterate. Unfortunately, since we are dealing with a constrained optimization problem it is not sufficient just to compare objective values to decide upon the next current iterate. It is also important to take into account the feasibility of a design. The last evaluated design may have improved the objective value, but on the other hand also moved outside the feasible region. This yields two conflicting sources of information regarding the distance to the optimum. One way to deal with constraints is to accept only feasible designs as

current iterate. However, this approach forces the path towards the optimum to stay in the feasible region and may unnecessarily slow down the convergence of the algorithm. If an infeasible design is encountered it may very well be closer to the optimum than the old current iterate, and thus it should provide a promising step on the path towards the optimum. Hence, it seems sensible that the algorithm has to use multiple criteria based upon the objective value and the feasibility of the designs.

The algorithm selects the current iterate based upon the *filter method* introduced by Fletcher and Leyffer (2002). The filter method uses two criteria: the value of the objective function and the satisfaction of the constraints. Note that we have assumed that X is defined as the feasible region of constraints (4.1c) and (4.1d) which have to be satisfied to perform a black-box evaluation. Hence, regarding the constraint violations we can concentrate on constraints (4.1b). The two filter criteria are formally defined as follows. For $x \in X$ and $r \in \mathbb{R}^m$ define

$$z(x,r) := f_0(x,r), \tag{4.2}$$

$$h(x,r) := \sum_{i=1}^{m_f} \max\{-s_i^l(x,r),\ -s_i^u(x,r),\ 0\}, \tag{4.3}$$

where $s_i^l(x,r)$ and $s_i^u(x,r)$ are the slack variables corresponding to constraints (4.1b), i.e.,

$$\begin{aligned} s_i^l(x,r) &:= f_i(x,r) - l_i^f, && 1 \le i \le m_f, \\ s_i^u(x,r) &:= u_i^f - f_i(x,r), && 1 \le i \le m_f. \end{aligned}$$

Hence, $z(x,r)$ is the objective value of problem (4.1), and $h(x,r)$ is the sum of the constraint violations of constraints (4.1b). Thus, $h(x,r(x)) = 0$ means that design x satisfies (4.1b).

Assume that at the current stage of the algorithm k design vectors $x^{(1)}, \ldots, x^{(k)}$ have been evaluated. Let $z^{(i)}$ and $h^{(i)}$ denote the values of $z(x^{(i)}, r(x^{(i)}))$ and $h(x^{(i)}, r(x^{(i)}))$ $(1 \le i \le k)$. The filter method uses the concept of domination for the objectives z and h.

Definition 4.1 *A pair $(z^{(i)}, h^{(i)})$ is said to* dominate *another pair $(z^{(j)}, h^{(j)})$ if and only if both $z^{(i)} \le z^{(j)}$ and $h^{(i)} \le h^{(j)}$.*

Applying this domination principle on a set of pairs leads to the formation of a filter:

Definition 4.2 *A* filter *is a list of pairs $(z^{(i)}, h^{(i)})$ such that no pair dominates any other. A pair (z,h) is said to be acceptable for inclusion in the filter if it is not dominated by any other point in the filter.*

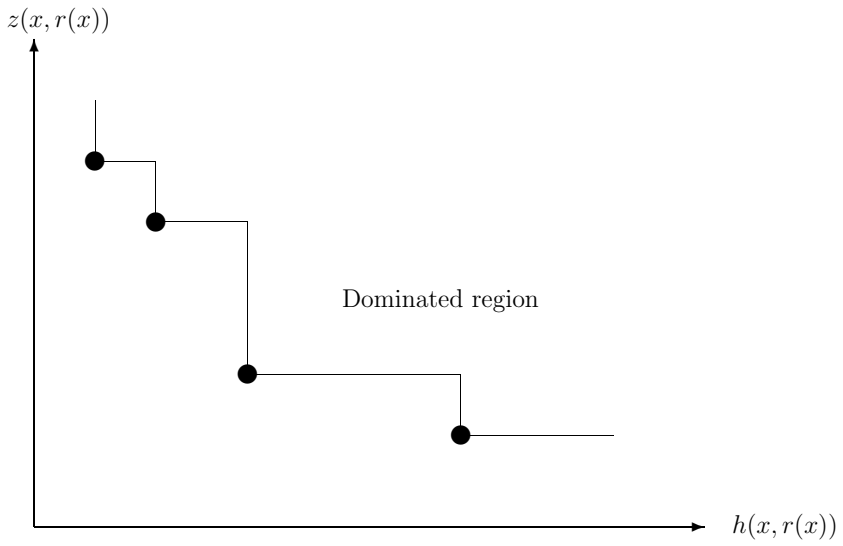

Figure 4.2: Example of a filter.

The filter can be graphically represented in the (z, h)-space as illustrated by Figure 4.2. Each point dominates the points located in the upper-right area from that point.

From here on we shall say that a design dominates another design, or that a design is accepted in the filter, as a shorthand of referring to the (z, h)-pairs of corresponding designs.

The basic principle of the use of a filter for the selection of the current iterate is that the last evaluated design becomes the current iterate if it is accepted in the filter. If the design is dominated by any of the previously evaluated designs, then the current iterate remains unchanged. Hence, the current iterate is defined as the last accepted entry into the filter. We have one exception to this rule, namely immediately after the initialization phase we select the start design $x^{(0)}$ if its feasible, or the most feasible filter entry otherwise.

As mentioned by Fletcher and Leyffer (2002) several problems can arise with the use of the filter method as outlined above. To overcome these problems Fletcher and Leyffer (2002) present some refinements in the filter's acceptance criteria. These problems and corresponding refinements have proved relevant to the SEQUEM algorithm as well.

A first problem that can be encountered is that while we are ultimately interested in a feasible solution for problem (4.1) the filter does not prevent the inclusion of an infinite sequence of designs for which $z^{(i+1)} < z^{(i)}$ and $h^{(i+1)} > h^{(i)}$ with $h^{(i)} \to \infty$. This situation can easily be excluded by adding the condition $h^{(i)} < h_{\max}$ for the inclusion in the filter, with $h_{\max} > 0$. An easy way to implement this condition is by adding an artificial pair

$(-\infty, h_{\max})$ to the filter. At the start of the algorithm we set $h_{\max} = \gamma m_f$, hence, a design for which all m_f constraints in (4.1b) are violated by an amount γ lies exactly on the upper bound $h_{\max}$. Note that during the course of the algorithm $h_{\max}$ can be decreased to force the algorithm towards feasible solutions.

A second problem that can occur is that the path from the current iterate to the optimum is blocked by one of the previously evaluated designs, a so-called *blocking entry*. Of course, the basic idea of the filter method is to reject a pair $(z^{(k)}, h^{(k)})$ that is dominated by another pair. However, note that two filter pairs $(z^{(i)}, h^{(i)})$ and $(z^{(j)}, h^{(j)})$ can be close together while corresponding designs $x^{(i)}$ and $x^{(j)}$ are located far apart. Therefore, a new evaluated design inside the trust region around the current iterate, say $x^{(i)}$, can be blocked from the filter by the design $x^{(j)}$. This can be especially annoying when the entire trust region is located in the infeasible region of problem (4.1) thereby preventing the algorithm to find a route back to the feasible region. Consequently, if such a situation is recognized, then the blocking entry is removed from the filter. Once a blocking entry is removed, it can never return in the filter in the remainder of the algorithm.

A possible third problem is the convergence to a pair (z, h) with $h > 0$. As explained by Fletcher and Leyffer (2002) this can be prevented by producing an envelope below the filter that prevents points arbitrarily close to the filter from being accepted. In all test problems that we have used to test the algorithm we have not encountered this problem. This is possibly due to the fact that $h_{\max}$ is decreased in certain stages of the algorithm forcing the designs towards the feasible region. Hence, to keep the algorithm as simple as possible the algorithm imposes no additional restrictions to avoid marginal filter improvements.

We are now ready to formulate the exact selection method for the current iterate. It is assumed that the evaluated designs, which are the candidates for the next current iterate, are denoted by $x^{(1)}, \ldots, x^{(k)}$. Furthermore, there is an upper bound $h_{\max}$ for the constraint violation function $h(x, r)$, and the indices of the blocking entries which are banned from the filter are given by the set $\mathcal{B}$. The decisions regarding $h_{\max}$ and $\mathcal{B}$ are discussed in Section 4.3.7. The set of indices that correspond to filter pairs after k evaluations is given by

$$\mathcal{F}_k := \big\{ i \in \{1, \ldots, k\} \setminus \mathcal{B} \;\big|\; h^{(i)} < h_{\max}, \\ \nexists j \in \{1, \ldots, k\} \setminus \mathcal{B} \text{ such that } (z^{(j)}, h^{(j)}) \text{ dominates } (z^{(i)}, h^{(i)}) \big\}.$$

The current iterate now corresponds with the last point accepted in the filter, i.e., $x^{(c)}$ with $c = \max_{i \in \mathcal{F}_k} i$.

4.3.4 Approximation of local model

An important part of the algorithm is the approximation of the black-box responses $r(x)$. It is our aim to approximate the actual response function $r(x)$ in the neighborhood of the current iterate $x^{(c)}$, more precisely within the trust region. Two conflicting aspects play a role here: the desire of high quality approximations and the time/cost involved by the black-box evaluations needed to fit the model. Obviously, high quality approximations are desirable. However, accurate approximations usually require a complex model specification which, in turn, requires many black-box evaluations to estimate the model parameters.

In our algorithm linear approximations are used to model the responses $r(x)$, i.e., for each response variable $r_j(x)$ $(1 \leq j \leq m)$ we specify the first order polynomial model

$$r_j(x) = (1\ x^T)\beta^{(j)} + e^{(j)}, \qquad x \in \mathbb{R}^n,$$

where $\beta^{(j)} \in \mathbb{R}^{n+1}$ contains the $n+1$ parameters of the model, and $e^{(j)}$ is the error term.

A disadvantage of the linear model is that it cannot properly approximate the nonlinear function $r_j(x)$. However, if the trust region is small enough, then the response $r_j(x)$ can be approximated locally with a linear function. Moreover, the main objective is not the accurate approximation of $r(x)$, but finding the location of the optimum of problem (4.1). Since the linear model requires only $n+1$ observations to fit the model the linear model saves black-box evaluations which can be used to explore the design space leading to the solution of problem (4.1). Note that after the first approximation of the local model, each time the local model is approximated at most one new black box evaluation has taken place, i.e., the designs evaluated in earlier stages are also used in the approximation.

Suppose that k design vectors $x^{(1)}, \ldots, x^{(k)}$ and corresponding observations $r(x^{(1)}), \ldots, r(x^{(k)})$ are obtained so far. For notational purposes we use the matrices

$$D := \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(k)} & \cdots & x_n^{(k)} \end{pmatrix} \tag{4.4}$$

and

$$R := \begin{pmatrix} r_1(x^{(1)}) & \cdots & r_m(x^{(1)}) \\ \vdots & \ddots & \vdots \\ r_1(x^{(k)}) & \cdots & r_m(x^{(k)}) \end{pmatrix}.$$

Let the current trust region be given by $TR(x^{(c)}, \Delta)$, where $x^{(c)}$ is the current iterate which is determined as explained in Section 4.3.3. Recall that $x^{(c)}$ is one of the design vectors evaluated so far. By the nature of the algorithm we are only interested in approximating

$r(x)$ within the trust region. Therefore, the parameters of the linear model are estimated using Ordinary Least Squares (OLS) with constant weights. The weight of observation i, w_i, is chosen equal to 1 if observation i has to be used to fit the linear model, and 0 otherwise. The OLS problem for $r_j(x)$ $(1 \leq j \leq m)$ then becomes

$$\min_{\beta^{(j)} \in \mathbb{R}^{n+1}} \sum_{i=1}^{k} w_i \big(r_j(x^{(i)}) - (1\ x^{(i)T})\beta^{(j)}\big)^2, \tag{4.5}$$

or in vector notation

$$\min_{\beta^{(j)} \in \mathbb{R}^{n+1}} \big(R_j - D\beta^{(j)}\big)^T W \big(R_j - D\beta^{(j)}\big),$$

where R_j denotes the j-th column of R, and $W \in \mathbb{R}^{k \times k}$ is a diagonal matrix with the weights $w_1, \ldots, w_k$ on its main diagonal.

As explained before, the linear model may have difficulties approximating the responses when these behave nonlinearly inside the trust region. Therefore, it may be desirable to let the approximation be exact in the current iterate. Hence, to yield this property the least squares problem in (4.5) could be extended with the constraint

$$r_j(x^{(c)}) = (1\ x^{(c)\,T})\beta^{(j)}. \tag{4.6}$$

In the selection of the weights we restrict ourselves to 0/1-type weights. Hence, the procedure explained above is equal to applying standard least squares using only a subset of the complete set of observations. Note that least squares is by no means limited to the use of 0/1-type weights.

It remains to decide whether an observation i should be included in the subset used to fit the linear model. Since we want to obtain good approximations inside the trust region it is obvious that all designs inside the trust region should be included. However, a design that is just outside the trust region could provide useful information for the approximation at the boundary of the trust region. Therefore, in principle, only designs $x^{(i)}$ that are inside the *weight-region*, a blown-up version of the actual trust region $TR(x^{(c)}, \Delta)$, receive weight 1. Thus,

$$w_i = \begin{cases} 1 & \text{if } x^{(i)} \in TR(x^{(c)}, \omega\Delta), \\ 0 & \text{if } x^{(i)} \notin TR(x^{(c)}, \omega\Delta), \end{cases} \tag{4.7}$$

where $\omega \geq 1$ is a parameter that specifies relative size of the weight-region to the trust region. This weight selection is illustrated by Figure 4.3. The designs inside the weight-region receive weight 1 are marked with a filled dot, and the designs outside the weight-region receive weight 0 are marked with an open dot.

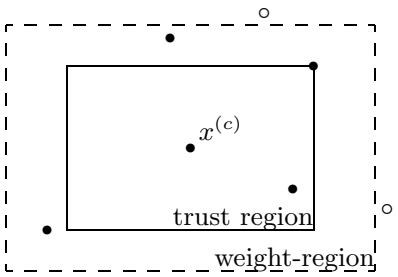

Figure 4.3: The trust region and the weight-region.

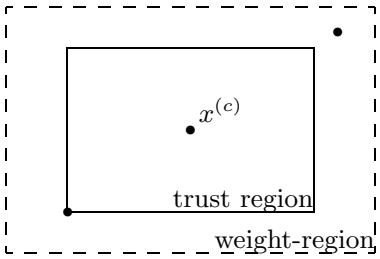

Figure 4.4: Linearly dependent designs.

Unfortunately, in some cases using only the weight formula in (4.7) is not applicable as there is an extra condition that the weights $w_1, \ldots, w_k$ have to satisfy. The weighted least squares solution of (4.5) is not uniquely determined if the matrix $D^T W D$ is singular. A unique solution exists if the $(\sum_{i=1}^{k} w_i) \times (n+1)$ matrix $\tilde{D}$, whose rows are given by the rows of D that correspond to weights $w_i = 1$, has linearly independent columns. It immediately follows that we need $\sum_{i=1}^{k} w_i \geq n+1$, which, unfortunately, is not sufficient to guarantee non-singularity. This is illustrated by Figure 4.4 where the three designs inside the weight-region are linearly dependent. In a case like this it is required to add a design outside the weight-region to the subset of observations determined by (4.7). The selection of the design $x^{(i)} \notin TR(x^{(c)}, \omega\Delta)$ that has to be added is determined by the distance to the current iterate $x^{(c)}$. The design that is closest to $x^{(c)}$ will be given $w_i = 1$. This procedure is then repeated until the design matrix $\tilde{D}$ has linearly independent columns.

4.3.5 Computing filter improving designs

The approximations of the responses computed in Section 4.3.4 are used to move towards the optimum of problem (4.1). We do this by substituting the linear approximations denoted by $\hat{r}_j(x)$ $(1 \leq j \leq m)$ into (4.1). Since the approximations are assumed only to approximate the actual responses within the trust region we add the restriction $x \in TR(x^{(c)}, \Delta)$. This yields the problem

$$\min z = f_0(x, \hat{r}(x)) \tag{4.8a}$$

$$\text{s.t. } l_i^f \leq f_i(x, \hat{r}(x)) \leq u_i^f, \qquad 1 \leq i \leq m_f, \tag{4.8b}$$

$$l_j^g \leq g_j(x) \leq u_j^g, \qquad 1 \leq j \leq m_g, \tag{4.8c}$$

$$l^x \leq x \leq u^x, \tag{4.8d}$$

$$x \in TR(x^{(c)}, \Delta). \tag{4.8e}$$

This problem is now explicitly known and can be solved using an NLP solver.

Firstly, suppose that problem (4.8) is feasible and let x^* denote its solution. In Section 4.3.3 we have selected the current iterate which is the design that is expected to be closest to the optimum of problem (4.1) based upon the filter method. Hence, the filter concept can also be used to compute filter improving designs instead of a feasible design alone. The local solution x^* has an expected location $(z(x^*, \hat{r}(x^*)), 0)$ in the filter space. In general, every design $x \in TR(x^{(c)}, \Delta)$ corresponds to an expected pair $(z(x, \hat{r}(x)), h(x, \hat{r}(x)))$. By allowing a certain constraint violation it may be possible to yield a lower expected objective value than $z(x, \hat{r}(x))$. The most efficient design w.r.t. the filter criterion that yields an expected objective value of at most z is given by the solution of the problem

$$\begin{aligned}
&\min\ h(x, \hat{r}(x)) && && \text{(4.9a)}\\
&\text{s.t. } z(x, \hat{r}(x)) \leq z, && && \text{(4.9b)}\\
&\quad l_j^g \leq g_j(x) \leq u_j^g, && 1 \leq j \leq m_g, && \text{(4.9c)}\\
&\quad l^x \leq x \leq u^x, && && \text{(4.9d)}\\
&\quad x \in TR(x^{(c)}, \Delta). && && \text{(4.9e)}
\end{aligned}$$

Solving (4.9) for several different values of z yields a set of designs that could provide promising candidates with respect to the (expected) filter improvement. Sensible objective targets z should be below $z(x^*, \hat{r}(x^*))$, and above the objective value that results from solving (4.8) without constraint (4.8b).

Secondly, if the local problem (4.8) is infeasible, then a first alternative is to minimize the constraint violations. Focusing on constraints (4.8b) this comes down to solving (4.9) without constraint (4.9b). Just like the case where the local problem is feasible this yields an upper bound for the objective value that can be used to solve (4.9). Hence, the same approach as above can be used if the local problem is infeasible.

For numerical reasons the objective in (4.9a) is very nasty as it holds all the approximated constraints (4.1b) into a single value. By introducing m_f extra decision variables

we can overcome this problem by reformulating problem (4.9) to

$$\min \sum_{i=1}^{m_f} h_i \tag{4.10a}$$

$$\text{s.t. } z(x, \hat{r}(x)) \leq z, \tag{4.10b}$$

$$h_i \geq -s_i^l(x, \hat{r}(x)), \qquad 1 \leq i \leq m_f, \tag{4.10c}$$

$$h_i \geq -s_i^u(x, \hat{r}(x)), \qquad 1 \leq i \leq m_f, \tag{4.10d}$$

$$h_i \geq 0, \qquad 1 \leq i \leq m_f, \tag{4.10e}$$

$$l_j^g \leq g_j(x) \leq u_j^g, \qquad 1 \leq j \leq m_g, \tag{4.10f}$$

$$l^x \leq x \leq u^x, \tag{4.10g}$$

$$x \in TR(x^{(c)}, \Delta). \tag{4.10h}$$

It should be mentioned that solving several instances of problem (4.10) in addition to problem (4.8) is computationally intensive. However, these computations can be useful as they might save some much more costly black-box evaluations. Note that the computation of the set of filter improving designs does not involve any black-box evaluation. The actual selection of the design to be evaluated by the black-box machine is discussed in Section 4.3.7.

4.3.6 Improving the geometry

In Section 4.3.4 it is explained that the designs used to approximate the responses have to satisfy a certain condition to be able to estimate a linear model using least squares. Unfortunately, this condition is not sufficient to guarantee accurate approximations. There are two important reasons why approximations may be lacking accuracy. Firstly, the actual behavior of the responses within the trust region does not match a linear model well enough. This does not necessarily have to be problematic if the approximations do result in improvements with respect to the filter criterion. However, if it is not possible to find improvements, then it is desired to decrease the trust region such that the linear model can approximate the responses more accurately.

Unfortunately, there is a second reason why the quality of the approximations can be insufficient to find improving designs. This situation occurs when the designs used to fit the model are located in such a way that they do not contain enough information to estimate the parameters of the linear model. The designs are then said to have a bad *geometry*. Usually, this happens when there is little variance in one or more of the directions of the design space. This is illustrated by Figure 4.5 where the designs in the weight-region are nearly located on a straight line. Unlike the designs in Figure 4.4 they are not linearly dependent, and can be used to fit a linear model using least squares.

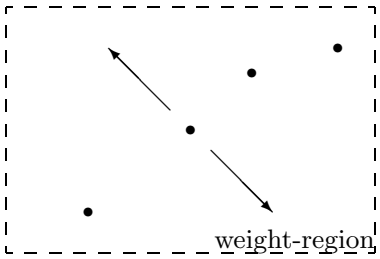

Figure 4.5: Bad geometry.

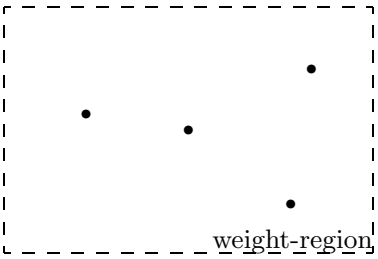

Figure 4.6: Good Geometry.

However, approximations in the directions given by the arrows in Figure 4.5 are prone to result in significant errors.

A set of designs has a good geometry if it fills the design space properly in all directions of the design space. An example of a good geometry is shown in Figure 4.6.

As a measure of the geometry we use a concept from the field of the Design of Experiments (DoE). In Driessen *et al.* (2006) it is shown that this is an attractive measure for solving black-box optimization problems. In Section 4.3.2 we already used a D-optimal DoE for the creation of the initial set of designs. A D-optimal design maximizes the determinant of D^TD, with D, the design matrix of the model, defined by (4.4). Since we do not use all available designs we concentrate on the weighted version: D^TWD.

For $x \in \mathbb{R}^n$ let $D(x) \in \mathbb{R}^{(k+1)\times(n+1)}$ be the design matrix for a linear model with all evaluated designs $x^{(1)}, \ldots, x^{(k)}$ extended with x, i.e.,

$$D(x) := \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(k)} & \cdots & x_n^{(k)} \\ 1 & x_1 & \cdots & x_n \end{pmatrix}.$$

Augmentation of an existing set of designs with a new design x inside the trust region using the D-optimality criterion yields the objective $\det(D(x)^TD(x))$. The procedure used to approximate the local model explained in Section 4.3.4 prevents the straightforward application of optimizing the objective mentioned above. This objective is based upon the assumption that all available designs will be used for the next regression. In our case this is not true, and, even worse, only until a new design has been evaluated it is possible to compute the weights that determine which subset of the evaluated designs is used to fit the model. If another current iterate is selected, then automatically the weight-region is moved accordingly. Even if the current iterate remains the same after evaluating a new design, then this might have consequences for some of the other designs' weights. Naturally, since the new design is chosen inside the trust region it is given weight 1. However, if the previous computation of the weights required adding some

extra designs outside the weight-region to guarantee non-singularity of D^TWD, then this may not longer be necessary in the new situation.

For the computation of geometry improving designs it is assumed that the current iterate does not change. If this assumption turns out to be invalid, then this implies that an improvement has been made in the filter sense. This can be seen as a stroke of good fortune, since the only reason to perform a geometry improving step is the lack of confidence in the ability of the current model to yield a filter improving step. The advantage of knowing that the current iterate does not change is that the new weight matrix $W(x)$ after adding design x can be computed beforehand. The best geometry improving design then follows from the problem

$$\max_{x \in X} \det\big(D(x)^T W(x) D(x)\big) \tag{4.11a}$$

$$\text{s.t. } l_j^g \leq g_j(x) \leq l_j^u, \qquad 1 \leq j \leq m_g, \tag{4.11b}$$

$$l^x \leq x \leq u^x \tag{4.11c}$$

$$x \in TR(x^{(c)}, \Delta). \tag{4.11d}$$

Dykstra (1971) has shown that the determinant in (4.11a) can be replaced by a numerically more efficient quadratic function. This function is given by $x^T(D^TW(x)_{-(k+1)}D)^{-1}x$, where $W(x)_{-(k+1)}$ is the weight matrix after adding design x with the last row and column deleted. Note that this method requires that $D^TW(x)_{-(k+1)}D$ is non-singular.

Another special case of the geometry objective function in (4.11a) is when exactly $n+1$ weights on the diagonal of $W(x)$ are equal to 1. Let $\tilde{D}(x)$ denote the matrix whose rows correspond the designs that have weight equal to 1. Since $\tilde{D}(x)$ is a square $(n+1)\times(n+1)$ matrix, (4.11a) simplifies to $\det(\tilde{D}(x))^2$. Hence, the objective is simply to maximize the determinant of $\tilde{D}(x)$ which is a linear function of x.

In any other case, it is not possible to use a simplification of the geometry objective (4.11a). Hence, the rather more difficult optimization problem (4.11) has to be solved to obtain a a geometry improving design.

In general, problem (4.11) has a convex objective function which makes it difficult to find a global maximum to the optimization problem. A standard NLP method can be used to solve (4.11), however this does not guarantee a global optimum. By using several starting points one can increase the probability of finding the global optimum. Moreover, the local optima of (4.11) which result from this method can be useful as well. With respect to the geometry measure they might be only slightly worse than the global optimum, whereas for the objective and constraints of problem (4.1) they can be much better. Therefore, all local optima of (4.11) are saved as possible candidates for the next black-box evaluation.

4.3.7 Evaluate candidate or decrease trust region

In Sections 4.3.5 and 4.3.6 it is explained how to compute candidates for the next black-box evaluation. The final step of the iterative algorithm is either to select one of the candidates to be evaluated by the black-box machine or to decrease the trust region. In the remainder of this section we do not make any difference between candidates that are a result of the filter improving step, or the geometry improving step. Instead, each candidate is judged on the basis of a number of criteria, and the decision is based thereupon. Moreover, in addition to the individual criteria for each of the candidates the decision depends on a number of general criteria, which are equal for all candidates.

Before we proceed with the detailed description of the selection procedure we introduce some notation. Let k denote the total number of black-box evaluations carried out by the algorithm so far. The set of designs that have been evaluated so far is denoted by $\mathcal{H} = \{x^{(1)}, \ldots, x^{(k)}\}$. The collection of candidates for the next black-box evaluation is denoted by $\mathcal{C}$. Next, the individual criteria for the candidate set are discussed.

Relative geometry measure

A *relative geometry measure* is computed for all candidates. An absolute geometry measure for design $y \in \mathcal{C}$, which is also used in the geometry improving step, is given by $\det(D(y)^T W(y) D(y))$ as in (4.11a). The value of this determinant is heavily influenced by the dimension n and the choice of weights $W(y)$, and, consequently, it is not immediately recognized whether it indicates a good or a bad geometry. Let $y^{(g)}$ denote the candidate that has the highest value for this absolute measure. The relative geometry measure of a candidate $y \in \mathcal{C}$ now results from the quotient

$$\gamma(y) = \frac{\det(D(y)^T W(y) D(y))}{\det(D(y^{(g)})^T W(y^{(g)}) D(y^{(g)}))}.$$

Obviously $\gamma(y^{(g)}) = 1$. Furthermore, a small value of $\gamma(y)$ denotes a geometry that is much worse than the geometry corresponding to $y^{(g)}$. Since we are not interested in immediately rejecting all candidates other than $y^{(g)}$ a candidate $y \in \mathcal{C}$ is said to be acceptable w.r.t. the geometry if $\gamma(y) \geq \gamma^*$ for a certain γ^*.

Distance to previous designs

It is desirable that each design that is evaluated by the black-box provides additional information about the behavior of the responses in the neighborhood of the trust region. A design that is very close to a design that has already been evaluated is unlikely to yield more information than a design that has a larger distance to each of the already evaluated designs. Therefore, the candidates are judged by the *smallest distance* to any

of the already evaluated designs. Because we have to relate the additional information to the size of the trust region, the distances are computed after scaling with the current trust-region size $\Delta^{(k)}$, i.e.,

$$\delta(y) = \min_{x \in \mathcal{H}} \|\text{diag}(\Delta^{(k)})^{-1}(y - x)\|, \qquad y \in \mathbb{R}^n.$$

Any candidate $y \in \mathcal{C}$ with $\delta(y) < \delta^*$, for a fixed value of δ^*, is said to be too close to an old design and is rejected for evaluation by the black-box machine during this iteration. Note that this does not exclude y for black-box evaluation in any of the next iterations of the algorithm after the trust region has been decreased.

In general, designs with a small value for $\delta(y)$ are not particularly good w.r.t. the geometry measure $\gamma(y)$ as well. However, if the distribution of the designs in the neighborhood of the trust region is reasonably well, then adding a design y close to one of the old designs will not necessarily result in a failure of the geometry criterion. Therefore, a distance related measure is needed in addition to the geometry measure $\gamma(y)$.

Expected filter improvement

The previous two criteria only depend on the positions of the designs and the trust region in the design space X. There is need for another criterion that evaluates the (expected) value of a design w.r.t. problem (4.1). In line with the approach taken previously, this is done by combining the local model and the filter method. As explained in Section 4.3.5 a design $y \in \mathcal{C}$ corresponds to an expected location $(z(y, \hat{r}(y)), h(y, \hat{r}(y)))$ in the (z, h)-space. If this expected filter pair is not dominated by the current filter, then this pair adds a certain area to the dominated region of the filter as illustrated by Figure 4.7.

The improvement in filter sense of a design $y \in \mathcal{C}$ is measured by the area $\omega(y)$ that this design adds to the dominated region of the current filter. If $\omega(y)$ is large, then y is expected to yield a big filter improvement which makes it an attractive design for the next black-box evaluation.

Note that unfair comparison due to the existence of pairs beyond the extreme points of the current filter can be prevented by adding the artificial filter pairs $(-\infty, h_{\max})$ and $(z_{\max}, 0)$ to the current filter, with $z_{\max}$ a practical maximum value for the objective function, and $h_{\max}$ as explained in Section 4.3.3.

If we summarize the criteria above, then it can be said that we require a minimum standard w.r.t. the geometry and distance criteria measured by $\gamma(y)$ and $\delta(y)$, respectively, and the best possible expected filter improvement measured by $\omega(y)$. Hence, the *best candidate* according to these three criteria is given by

$$y^* = \text{argmax}_{y \in \mathcal{C}} \big\{ \omega(y) : \ \gamma(y) \geq \gamma^*, \ \delta(y) \geq \delta^* \big\}.$$

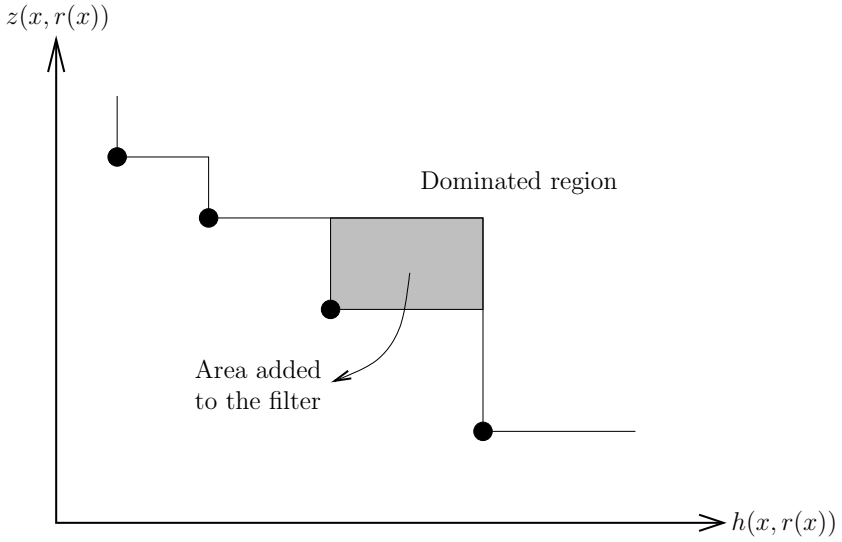

Figure 4.7: Extension of the dominated region by adding a filter pair.

It can be concluded that y^* is a very promising one among the candidates in $\mathcal{C}$. However, whether y^* will actually be evaluated also depends on other factors. In addition to the three individual measures above, which allow comparison between the candidates $y \in \mathcal{C}$, the decision taken in this step of the algorithm also depends on the following.

Current geometry

The *current geometry* plays an important role in deciding whether the trust region can be decreased or not. If the current geometry is not good, then it is not desirable to decrease the trust region since the current predictions of the local model might become more accurate if the geometry is improved. The geometry measure $\gamma(y)$ measures the effect of adding design y on the determinant relative to the best geometry design $y^{(g)}$; therefore, it does not give any information about the absolute effect on the determinant. The two determinants $\det(D^T W D)$ and $\det(D(y)^T W(y) D(y))$ together indicate the effect on the D-optimality criterion after adding a design y. Hence, if the quotient

$$\phi(y) = \frac{\det(D(y)^T W(y) D(y))}{\det(D^T W D)}$$

is large, then this indicates that the geometry measure can be significantly improved by adding the design y. For a good current geometry we require that the determinant

does not increase too much after adding the best geometry improving design $y^{(g)}$, i.e., $\phi(y^{(g)}) \leq \phi^*$, for a certain ϕ^*.

Even though the determinant is a good measure for the dispersion of the designs in the design space X, it does not take into account the position of the trust region. When comparing the possible candidates $y \in \mathcal{C}$ this is not much of a problem, because all candidates are located inside the trust region. However, if the trust region is decreased, then besides an evenly dispersion of the designs it is also desired that the designs are located near the trust region. Therefore, we impose two additional conditions on the current geometry. Firstly, the total number of designs inside the weight region has to be larger than a certain number: $|\{i : \ x^{(i)} \in TR(x^{(c)}, \omega\Delta)\}| > B$. Secondly, the weighted average of the designs has to be located reasonably close to the current iterate, i.e.,

$$\left\| \frac{\sum_{i=1}^{k} w_i x^{(i)}}{\sum_{i=1}^{k} w_i} - x^{(c)} \right\| \leq \eta,$$

for a certain value of η.

Decision scheme

Figure 4.8 shows the decision flow chart about the next action of the SEQUEM algorithm. The first item we are concerned with is the feasibility of the local model. If the local model is infeasible, i.e., problem (4.8) has no solution, then the SEQUEM algorithm initiates a *restoration step* which is intended to move the trust region back towards the feasible region of the original problem. It has been argued previously that it is not necessary to search for feasible designs all of the time. The filter concept provides an excellent method for finding a good balance between objective and constraints. However, if the local model is infeasible, then this could mean that the trust region is entirely located outside the feasible region of problem (4.1). In this case, we do not want to rely on the filter measure $\omega(y)$ for the selection of a design $y \in \mathcal{C}$, because aiming for a filter improvement possibly with a positive expected constraint violation could lead even further away from the feasible region. Therefore, the restoration step selects the most feasible solution that satisfies the geometry and distance criteria to be evaluated by the simulation tool, i.e.,

$$y^{(r)} = \text{argmin}_{y \in \mathcal{C}} \{h(y, \hat{r}(y)) : \gamma(y) \geq \gamma^*, \ \delta(y) \geq \delta^*\}.$$

If the local model is feasible, then the algorithm selects either y^* to be evaluated by the simulation tool, or the trust region will be decreased. If y^* is evaluated, then it will be labelled as either a *filter step* if we strongly expect y^* to be a filter improvement, or a *geometry step* otherwise.

This decision needs some further examination of the best candidate y^*. It is interesting to know whether there is a candidate that has a larger expected filter improvement, but

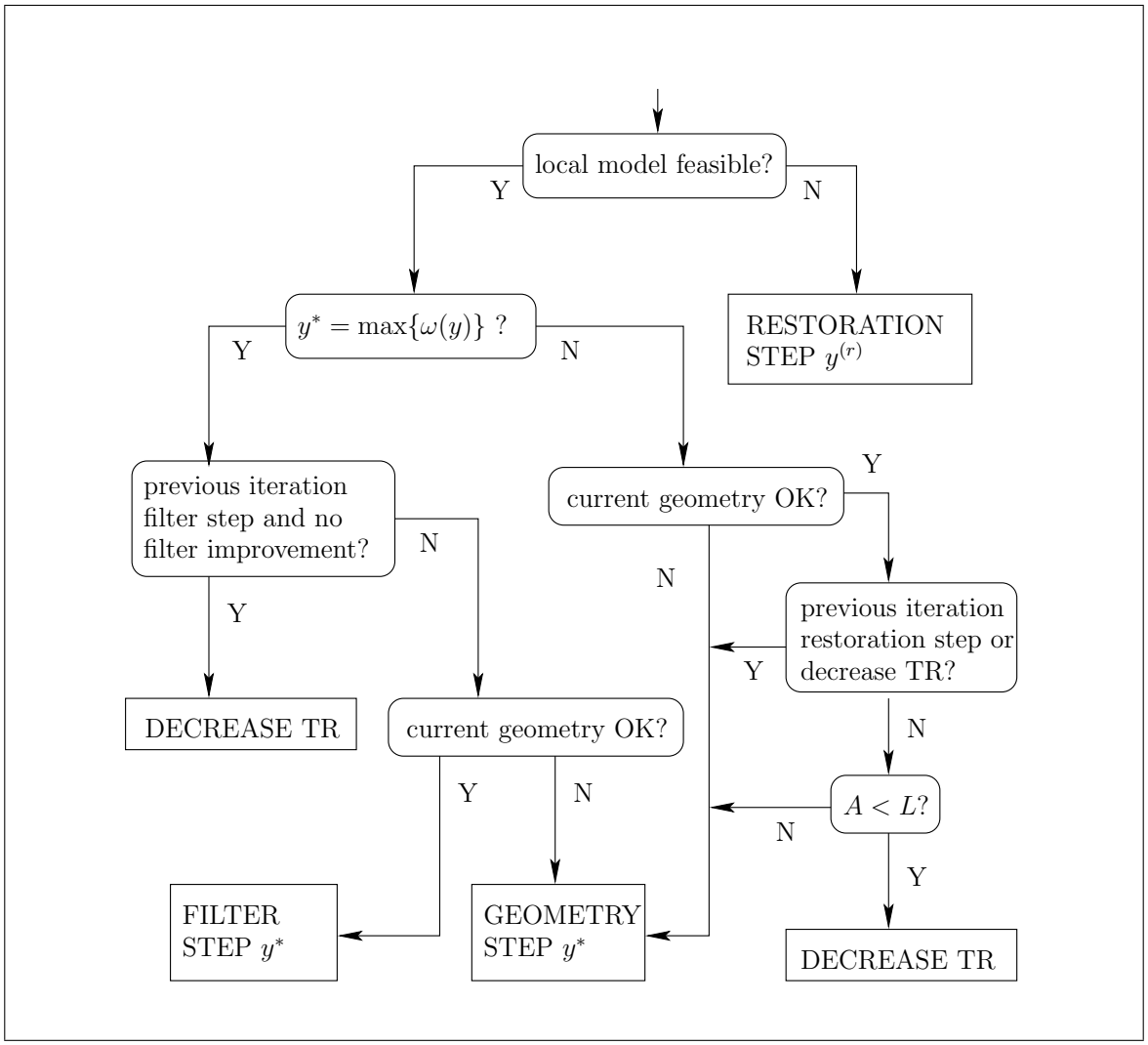

Figure 4.8: Decision flow chart.

which does not satisfy the geometry or distance criteria. If there is no such candidate, then this makes y^* an even more attractive candidate, possibly even a *filter step*. However, if in the previous iteration a filter step did not result in a filter improvement, then this probably means that we are searching in a trust region that is too large, since all other clues are pointing in the direction of filter improvement. If this is not the case, i.e., the previous iteration was not a filter step or the previous iteration was a filter step that resulted in a filter improvement, then there is one final check before we decide that y^* can be labelled a *filter step*. If the current geometry is not good, then the local model might not be accurate enough and it would not be right to initiate a filter step. In this situation a *geometry step* is executed. Alternatively, if the current geometry is good, then y^* is evaluated and labelled a *filter step*.

If there is a candidate that has a larger expected filter improvement than the best candidate, i.e., $y^* \neq \text{argmax}_{y \in \mathcal{C}}\{\omega(y)\}$, then we come to another branch in the decision scheme. The general question in this branch is whether we perform a *geometry step* by evaluating y^*, or whether we *decrease* the trust region.

If the current geometry is not good, then it is likely that we cannot find filter improvements because the local approximations are not accurate enough. Therefore, we perform a *geometry step* if the current geometry is not good.

We also perform a *geometry step* if, in the previous iteration, we either performed a *restoration step* or decreased the trust region. In the former case there is the danger that only a small part of the trust region is in the feasible region of the local model, hence decreasing the trust region may again lead to an infeasible local model in the next iteration. In the latter case the alternative would be to decrease the trust region in two consecutive iterations of the algorithm, which is undesirable because we still need to build good approximations in the new trust region.

Finally, we arrive at the decision point where the current geometry is good and the previous iteration was a filter or a geometry step. In this situation it is important whether a candidate $y \in \mathcal{C}$ with $\omega(y) > \omega(y^*)$ has been rejected because of the geometry criterion or the distance criterion. If one or more of these candidates only fail the geometry criterion, then this merely indicates that the geometry needs to be improved. In this case y^* is a good candidate since it satisfies the geometry criterion and also might yield nice filter results. If a number of candidates with a high expected filter improvement are rejected due to the distance criterion, then this indicates that the local model expects filter improvements for designs that very are close to already evaluated designs, which is not very probable. Therefore, the number of elements in the set $\{y \in \mathcal{C} : \omega(y) > \omega(y^*),\ \delta(y) < \delta^*\}$ is computed. If this number, say A, is smaller than a certain L, then we perform a *geometry step*. Otherwise, we *decrease the trust region* since the local model seems to be inaccurate even though the current geometry is good which indicates that we are near the optimum.

4.4 Preliminary numerical results

Before presenting some numerical results we first discuss the values for several tuning parameters. Default values used for the tuning parameters are $\gamma^* = \frac{1}{4}$, $\delta^* = \frac{1}{4}$, $\phi^* = 4$, $\eta = 2$, $\omega = 1\frac{1}{4}$, $B = 0.3n$, $L = \lceil 0.35n \rceil$. Of course, different values for these parameters may affect how the algorithm works. We advise to use these default values for "stand alone" problems. However, when the algorithm is repetitively used for a certain class of problems, then it is valuable to look for better settings for these parameters. A systematic way of finding good values for the parameters in such cases is to use DoE techniques on a test set for this specific class of problems. Moreover, when the budget (number of simulations) of the user is rather limited, then it might be better to use more aggressive values for the tuning parameters, e.g., to pay less attention to the geometry condition (i.e. use lower values for ϕ^*).

The SEQUEM algorithm has been implemented for the case where the functions $f_i (0 \leq i \leq m_f)$ and $g_j (0 \leq j \leq m_g)$ are assumed to be linear. Note that this assumption does not prohibit nonlinear behavior of the black-box process. Moreover, nonlinear constraints on the design parameters can be handled by the introduction of artificial response parameters representing the nonlinear terms of the constraints. The implementation of SEQUEM has been tested on a number of academic test problems.

Taking into account that the SEQUEM algorithm is specifically designed for optimization problems involving time consuming simulations, the relevant range of the number of design parameters that can be handled by the SEQUEM algorithm is approximately 10-50. This range matches with the size of relevant problems that we have experienced in practice.

We will test our algorithm on two theoretical and one practical problem. The first problem is the "cantilever beam" optimization problem from Toropov *et al.* (1993) given by

$$\begin{array}{rlll} \min\limits_{x \in \mathrm{R}^5} & r_1(x) & & \\ \text{s.t. } r_2(x) & \leq & 1, & \\ x_i & \geq & 0, & (1 \leq i \leq 5), \end{array}$$

where the response function $r : \mathbb{R}^5 \rightarrow \mathbb{R}^2$ is defined by

$$\begin{aligned} r_1(x) &= 0.0624(x_1 + x_2 + x_3 + x_4 + x_5), \\ r_2(x) &= \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3}. \end{aligned}$$

Results for SEQUEM for this problem were already reported in Brekelmans *et al.* (2001). The second, larger, test problem is represented by problem no. 284 (TP 284) from Schit-

tkowski (1987) which has 15 variables, 10 quadratic inequality constraints, and a linear objective function. We introduced one response parameter to capture the objective function, and one response parameters is introduced for each constraint as well. Hence, the black-box optimization problem has 15 design parameters and 11 response parameters. Problem TP 284 has been solved by the SEQUEM algorithm as well as by the FMINCON procedure from the MATLAB Optimization Toolbox (2000) and by COBYLA. The algorithm used by FMINCON is a Sequential Quadratic Programming (SQP) method using finite-differencing to estimate the gradients.

Figures 4.9 and 4.10 show the progress of the best feasible solution as a function of the number of black-box evaluations for both SEQUEM, FMINCON and COBYLA. For both problems, it can be seen that the objective function decreases much more rapidly for the SEQUEM algorithm than for FMINCON. The results are also in line with our expectation that the differences between SEQUEM and optimization methods which use finite-differences (like FMINCON) will be bigger for larger problems. For the second problem FMINCON makes one big step towards the optimum after about 650 evaluations. SEQUEM, on the other hand, reaches the theoretical optimum function value -1840 within 3% measured from the initial function value -522 after only 200 black-box evaluations. But the latter is not necessary a drawback because most practical problems involving time consuming simulations possess a certain degree of noise. Comparing SEQUEM and COBYLA, we conclude that for Problem 2 COBYLA performs slightly better, but for Problem 1 SEQUEM performs much better.

In the final part of this section we describe the application of the SEQUEM algorithm in a practical problem involving battery management (see Bergveld (2001)) at Royal Philips Electronics. Batteries have become very important for portable equipment such as telephones, computers and other similar devices. Various battery types are available for these purposes and their electrochemical description is a topic of research. In this setting we looked at the problem of finding optimal parameter settings for a simulation model for the process of recharging a battery. Our aim is to calibrate the simulation model to real data, collected by measuring the voltage and pressure during recharge at 900 successive moments in time.

We studied the case in which the simulation model requests as input the settings for 18 design parameters. Part of the research was to compare our sequential algorithm with the currently used MATLAB procedure LSQNONLIN. The objective function is given by (4.12).

$$F(x) = \sum_{i=50}^{950}((V_t(x) - V_t)/V_t)^2 + \sum_{i=50}^{950}((\log(P_t(x)) - \log(P_t))/\log(P_t))^2 \tag{4.12}$$

Here, $V_t(x)$ and $P_t(x)$ denote the simulated voltage and pressure at time t respectively, and

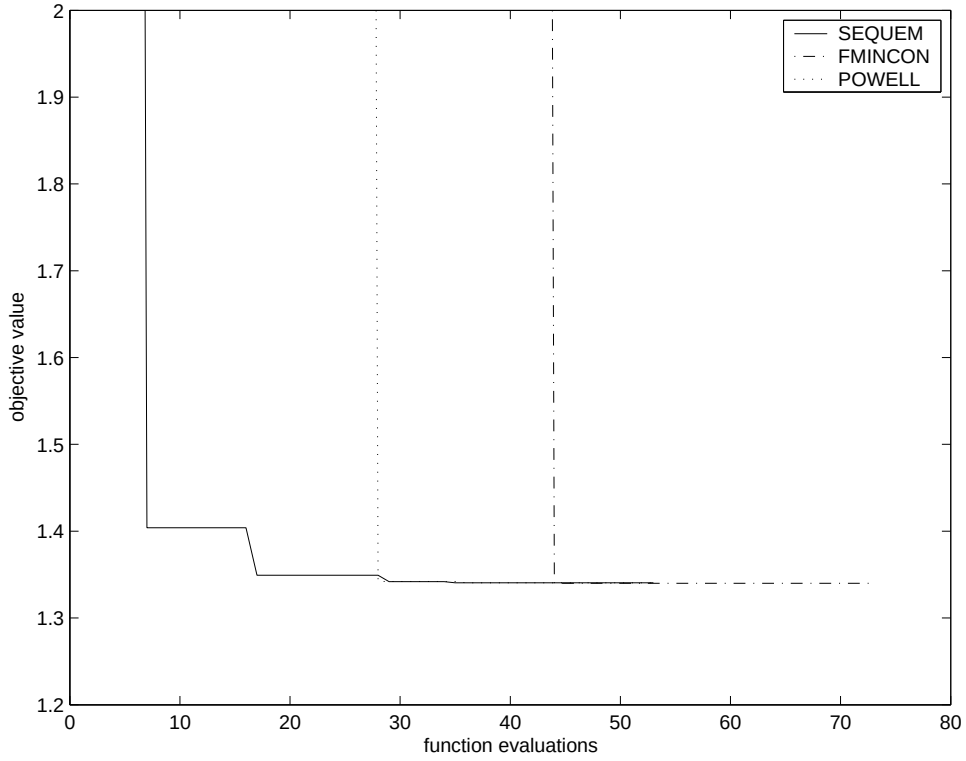

Figure 4.9: Best feasible solution for SEQUEM, COBYLA, and FMINCON as a function of the number of evaluations for the test problem "Cantilever Beam".

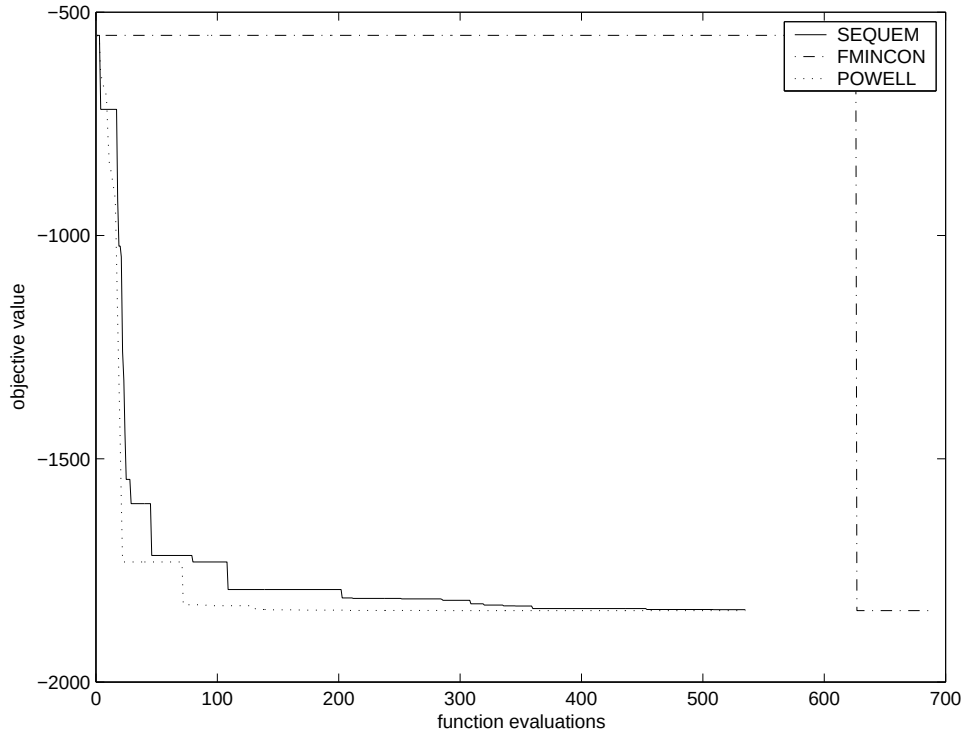

Figure 4.10: Best feasible solution for SEQUEM, COBYLA, and FMINCON as a function of the number of evaluations for test problem TP 284.

V_t and P_t denote the measured voltage and pressure respectively. Each design parameter is bounded by a simple lower and upper bound. The treated data set has been acquired at the Philips Research Laboratories of Eindhoven, The Netherlands, by measuring a NiCd P60AA Panasonic battery charged at a 1 C rate for 2 hours at 25 C ambient temperature.

We started LSQNONLIN in the center of the design region. After 1173 iterations the procedure terminated. The objective value had decreased from an initial value of 312.50 to 0.81. We also started our sequential algorithm in the midpoint of the design region. We stopped it after 750 simulations and by then it had reached an objective value of 0.49. Figure 4.11 shows the best objective value so far against the simulation number for LSQNONLIN (upper line) and SEQUEM (lower line). Again we can conclude that SEQUEM achieves large objective improvements rather quickly. In this specific example SEQUEM converged to a better optimum than the LSQNONLIN solver. We expect that this comes from the ability of SEQUEM to not get stuck in small local optima arising from numerical noise.

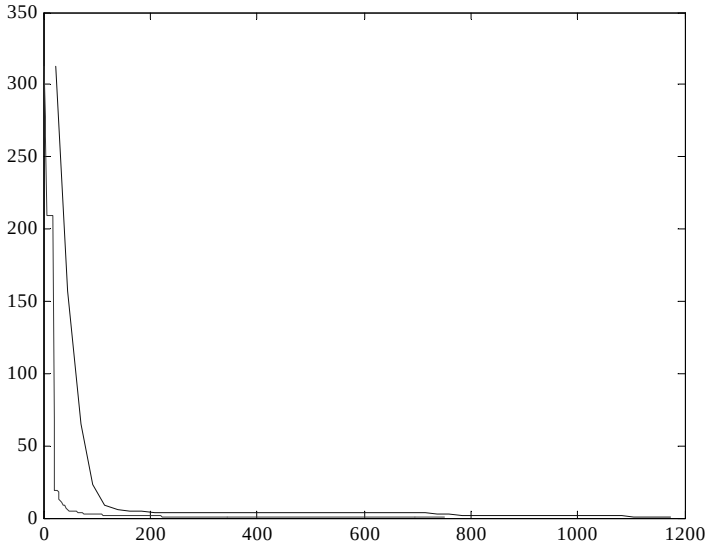

Figure 4.11: Results of SEQUEM vs LSQNONLIN.

4.5 Conclusions

Due to the developments in the area of CAE, computer simulations are used more frequently to evaluate and optimize products and processes. This leads to special nonlinear optimization problems in which the functions are not explicitly known and only time consuming simulations can be carried out to calculate the response function values. Most of the existing optimization methods require too many iterations, i.e. simulations.

The algorithm proposed in this paper tries to save simulation runs as much as possible by using local approximations which are also based on previous simulation runs. The quality of the approximations is safeguarded by a geometry measure for the locations of the simulation points. The use of a filter in the optimization algorithm provides the designer with a set of possible attractive designs to choose from instead of just a single optimal design. This set, given by the designs in the filter at the end of the algorithm, indicates the gain that can be reached w.r.t. the objective value at the cost of the constraint violations. This can be especially useful when it is possible to twiddle with the bounds of the constraints, and conventional sensitivity analysis is too costly due to the expensive function evaluations required.

This algorithm is implemented and preliminary computational results are promising. Finally we mention that also some of the new ideas from this paper can be combined with other (unconstrained) derivative-free algorithms. E.g. after some communication with

Marcelo Marazzi, we think that adding the geometry measures discussed in this paper will improve his algorithm described in Marazzi and Nocedal (2002). This is a subject for further research. Moreover, some parts of SEQUEM are used in an algorithm which is specifically developed for optimizing cooling strategies of electronic systems. For more details see Parry *et al.* (2004)

Chapter 5

On D-optimality based trust regions for black-box optimization problems

Abstract: Various sequential derivative-free optimization algorithms exist for solving black-box optimization problems. Two important building blocks in these algorithms are the trust region and the geometry improvement. In this paper we propose to incorporate the D-optimality criterion, well-known in design of experiments, into these algorithms in two different ways. Firstly, it is used to define a trust region that adapts its shape to the locations of the points in which the objective function has been evaluated. Secondly, it is used to determine an optimal geometry improving point. The proposed trust region and geometry improvement can both be implemented into existing sequential algorithms.

5.1 Introduction

Black-box optimization problems are common in, for example, engineering design optimization, where time consuming function evaluations are often carried out by simulation tools. For instance, Den Hertog and Stehouwer (2002a) discuss applications in the design of the color picture tube.

The unconstrained black-box optimization problem we consider, is stated formally as

$$\max_{d} \; f(d) + \epsilon \qquad d \in \mathbb{R}^q, \tag{P$_0$}$$

where q denotes the number of design parameters and ϵ denotes the error term. We assume that the error terms in (P$_0$) are i.i.d. with mean 0 and variance σ^2. The analytical form of the objective function f is unknown, as well as any derivative information. Hence, the only way to get information about this function is by evaluating it for distinct design points, or in short points. In this paper we assume that each function evaluation is expensive or time consuming. Therefore, keeping the number of function evaluations as low as possible is crucial. Most real-life applications involve constraints as well. The analysis of this paper can also be applied to constrained optimization problems. We focus on the unconstrained case for ease of notation.

Both in the fields Response Surface Methodology (RSM) (e.g., Myers and Montgomery (1995)) and Mathematical Programming sequential methods have been developed to solve

unconstrained black-box optimization problems.
Alexandrov *et al.* (1998) present a framework for generating a sequence of approximations to an expensive objective function based on pattern search methods. Toropov (1999) describes the use of a multipoint approximation method based on response surface fitting. Marazzi and Nocedal (2002) present a wedge trust region method for derivative free optimization. The derivative free methods by both Conn and Toint (1996) and Powell (2002) iteratively form quadratic models by interpolation and optimize these models in a trust region framework. An interesting aspect of these methods is that the points that determine the local approximations should satisfy some geometric conditions (e.g.,Powell (1994a), Conn and Toint (1996), Conn *et al.* (1997), Powell (2002)). These conditions ensure that the points used to determine the local approximations are well spread over all design dimensions in the neighborhood of the best point found so far. Both Powell (1994b) and Conn *et al.* (1997) use the determinant of a set of design points to obtain a good geometry. If these geometric conditions are not satisfied, a geometry improving step is carried out. Summarized, in all of these algorithms the trust region and the preservation of a good geometry play an important role. In the rest of this paper we will discuss the part of the sequential algorithms that takes care of the geometry conservation, the 'geometry improving step', and the part that takes care of the objective improvement, the 'objective improving step'.

A disadvantage of the generally used spherical trust region is that its location and shape are chosen independently of the location of the points on which the local models are fitted. Hence, if the design points are located in a long, narrow area, the trust region is still spherical, while it would be more accurate to apply a trust region with an adapted shape in this situation. Furthermore, using a spherical trust region makes the algorithms sensitive to an affine transformation (a transformation that is a combination of single transformations such as translation, rotation, or reflection on an axis) of the input variables. More precisely, the solution to the transformed problem and the transformation of the solution to the original problem are not necessarily the same. So the solution to the design problem in which the design variables are in meters can deviate from the solution to the same problem in which the design variables are in centimeters. This implies that, dependent on the chosen scaling, the algorithm evolves in a different way.

We suggest a new trust region that accounts for the disadvantages explained above. It automatically incorporates the information about the location of the points and is insensitive to affine transformations. This is achieved by linking the classical approach used in RSM and mathematical programming to the D-optimality criterion for finding good DoE schemes. This leads to the use of ellipsoidal trust regions in the objective improving step of the optimization approach. The position of the center of the ellipsoid as well as its shape depend on the location of the points on which the local models are

fitted. The D-optimality criterion is also incorporated in the geometry improving step. Some references for D-optimality are Box and Draper (1971) and Mitchell (1974). The suggested trust region and geometry improvement are possible alternatives for the current implementations of these concepts in existing sequential algorithms.

The rest of this paper is organized as follows. In Section 5.2 we introduce our new ellipsoidal trust region for the objective improving step of the optimization approach. Section 5.3 discusses the geometry improvement, also based on the D-optimality criterion. In Section 5.4 we show that use of our ellipsoidal trust region results in an objective improving step that is insensitive to affine transformations of the design space under consideration. Section 5.5 presents some illustrative examples of the use of the ellipsoidal trust region. The conclusions and future research are discussed in Section 5.6.

5.2 The ellipsoidal trust region in the objective improving step

In this section we take a close look at an optimization problem that has to be solved in the objective improving step of the optimization algorithm. We first formulate the classical trust region model. Then we give the intuition behind our new ellipsoidal trust region and formulate the resulting new model.

In general, determining the new evaluation point boils down to optimizing the approximating model under a trust region constraint. Some methods use linear approximating models, others partly or full quadratic models. We restrict our analysis to linear models, as these are often used in practical applications where the evaluation budget is very limited. The classical problem can then be stated as[1]

$$\begin{aligned} &\max_{d} \ \beta' d \\ &\text{s.t.} \left\| \ d - d^* \ \right\|_2 \leq \Delta, \end{aligned} \tag{P_1}$$

where $\|.\|_2$ denotes the 2-norm, the $'$ behind a vector or matrix denotes the transposed sign, $\beta \in \mathbb{R}^q$ is the vector of model coefficients arising, for example, from a linear least squares regression on the design points, $d \in \mathbb{R}^q$ is the decision variable, the vector of design parameters, $d^* \in \mathbb{R}^q$ is the best point found so far, and $\Delta \in \mathbb{R}$ is the trust region radius. Hence, a local linear model of the underlying real model is optimized within a spherical trust region.

The solution of (P_1) can be written down explicitly. By setting both the derivatives to d and to λ of the Lagrangian of problem (P_1) equal to zero and solving the resulting

[1]For ease of exposition the constant term has not been included in this formulation of the optimization model. Its presence does not alter the optimal solution of (P_1).

equations, we find that the optimal d becomes

$$d^{opt} = d^* + \frac{\Delta}{\| \beta \|_2}\beta. \tag{5.1}$$

The location of the points on which the local approximating models are fitted does not influence the shape of this spherical trust region in any way. Furthermore, in this trust region framework it is implicitly assumed that all design parameters are of comparable magnitude.

We believe that in the objective improving step, in which the aim is to find a locally optimal point with respect to the objective function given the data available at that moment, the trust region should be driven by the data points. The trust region is the region in which you trust your local models to approximate well. If in one dimension there is more information than in the other, and hence you trust the models to predict better in one dimension than in the other, then this should be reflected in the maximum stepsize you wish to take in those dimensions, hence in the shape of the trust region.

The dispersion of the design points contains information about the reliability of the models fitted on these points. In order to be able to incorporate this information into the method we formulate a new problem by changing the shape of the trust region. In statistics, a natural way to take locations of design points into account is by using the prediction variance of the approximation (see, e.g., Kleijnen *et al.* (2004)). Formally, this prediction variance is only valid if the true underlying function is linear. As the optimization approach consists of local approximations, the underlying function will be approximately linear if we look at a small enough scale.

As long as the variance remains within acceptable ranges, the model is trusted. The idea is to apply this approach to our problem. The variance is minimized in the center of gravity of the design points and the contour curves of this variance are ellipsoids. Before formulating this result formally in Theorem 5.1, we first introduce some notation. The results of Theorem 5.1 allow us to choose a suitable trust region during the optimization process.

In the statistic linear regression model the variance of the predictor $\hat{y} = \beta'_{+0}x$ equals $\sigma x'(X'X)^{-1}x$, where A^{-1} denotes the inverse of matrix A. With β_{+0} we denote the model coefficients including the coefficient for the constant term. The matrix X is known as the extended design matrix and consists of the row vectors $(x^i)' = [1\ (d^i)'], i = 1, \ldots, n$, where d^i denotes the design vector for the i^{th} experiment and n is the number of design points that are used for fitting the local approximations. The matrix X is assumed to have linearly independent columns. The symbol σ denotes the standard deviation of the error term. The covariance matrix $(X'X)^{-1}$ plays an essential role in D-optimality. We will show how this matrix induces a new trust region. As we focus on the design space

and do not take the constant term into account, we work with the matrix C instead of the matrix $(X'X)^{-1}$, which is related to $(X'X)^{-1}$ in the following way

$$(X'X)^{-1} = \begin{pmatrix} a & b' \\ b & C \end{pmatrix}, \tag{5.2}$$

where $a \in \mathbb{R}$, $b \in \mathbb{R}^q$, and $C \in \mathbb{R}^{q \times q}$. As $(X'X)^{-1}$ is positive definite and symmetric, C is positive definite and symmetric as well. Hence, C is also non-singular. Due to the special structure of the matrix X, the matrix $X'X$ has the following structure

$$X'X = \begin{pmatrix} n & n\bar{d} \\ n\bar{d} & D'D \end{pmatrix}, \tag{5.3}$$

where $\bar{d}$, the center of gravity of the design points $d^i, i = 1, ..., n$, is defined as

$$\bar{d} = \frac{1}{n} \sum_{i=1}^{n} d^i$$

and the matrix D equals X without the first column of ones, i.e., $X = [1\ D]$.

The following theorem, well known in statistics, shows that the ellipsoids arising from the matrix C are in fact contour curves of the variance.

Theorem 5.1 *The variance of the predictor $\hat{y} = \hat{\beta}'_{+0}x$ is minimal in $\bar{d}$, the center of gravity of the design points $d^i, i = 1, ..., n$. The contour curves of this variance are given by the ellipsoids*

$$(d - \bar{d})'C(d - \bar{d}) = \rho, \tag{5.4}$$

where $\rho = \rho_0 - a + \bar{d}'C\bar{d}$ and ρ_0 equals the variance of the predictor.

We propose to use the ellipsoids as defined in (5.4) in the definition of the trust region. The new problem formulation including the ellipsoidal trust region becomes

$$\begin{aligned} &\max_d \ \beta' d \\ &\text{s.t.} \left\| \ d - \bar{d} \ \right\|_C \leq \rho, \end{aligned} \tag{P_2}$$

where ρ is the trust region radius and $\| \ x \ \|_C$ is the C-norm defined by

$$\| \ x \ \|_C = \sqrt{x'Cx}.$$

As the matrix C is positive definite, it defines a proper norm. Also the solution of problem (P_2) is explicitly known. By setting both the derivatives to d and to λ of the Lagrangian of problem (P_2) equal to zero and solving the resulting equations, we find that the optimal d becomes

$$d^{opt} = \bar{d} + \frac{\rho}{\left\| \ \beta \ \right\|_{C^{-1}}} C^{-1}\beta. \tag{5.5}$$

The matrix C can be ill-conditioned and when solving the explicit solution one should take care not to compute this matrix by inversion, but to use for example the expression for C^{-1} that is derived in Theorem 5.2 to avoid numerical instability and loss of accuracy.

The first of the two main differences between problem (P_2) and problem (P_1) is that we now use the C-norm instead of the 2-norm. The second difference is that in problem (P_2) the center of the trust region is determined by all the design points on which the local linear models are fitted together, while in problem (P_1) the trust region is centered around the best point so far.

We illustrate the implications of using this C-norm instead of the 2-norm in Figure 5.1. Two important observations are illustrated in this figure. The first one is that the ellipsoidal trust region adapts its form to the locations of the design points, whereas the spherical trust region does not. This adaptation ensures that the models are more trusted in areas where actual evaluations have been performed. The second observation is that the center of the ellipsoidal trust region is determined by the design points such that the ellipsoid covers the design points in the best possible way. The spherical region is centered on the best point found so far. Hence, if such a point lies a bit apart from the other design points, some parts of the spherical trust region might not contain design points at all.

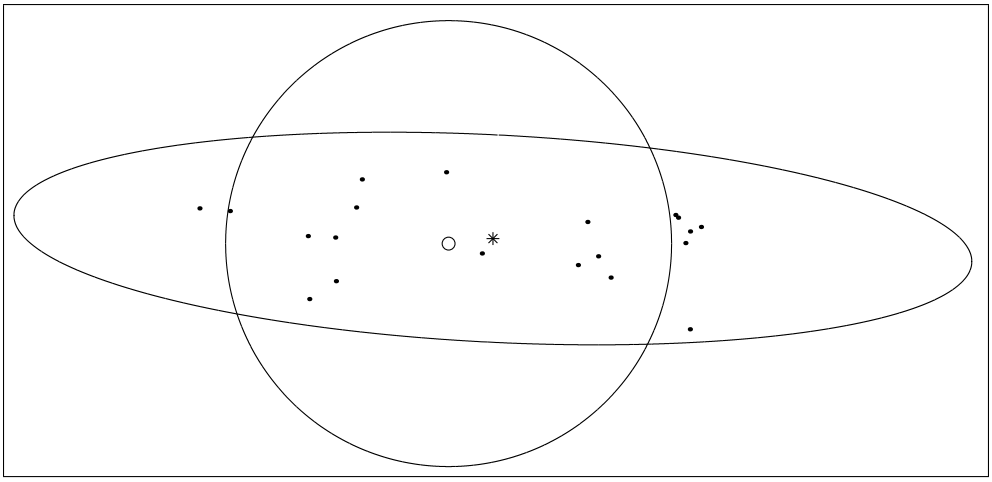

Figure 5.1: The ellipsoidal trust region adapts better to the locations of the design points than the spherical trust region. The small black dots indicate design points, the $*$ indicates the center of the ellipse, and the open dot is the center of the sphere.

When the ellipsoidal trust region becomes too narrow in one or more directions, this is an indication that the consecutively simulated design points have more or less the same value for these dimensions. Eventually, the approximating models will start to show lack of fit in these dimensions. In the next section we describe how to prevent the occurrence of this situation by means of a geometry improving step, that is also based on the same

matrix C.

The natural question arises how the use of higher-order approximation models would affect the above analysis. Let us consider quadratic models. For a quadratic model the extended design matrix X is extended with the second-order terms. The resulting matrix C then gives rise to a non-convex trust region and the equivalent of (P_2) becomes a non-convex NLP, which is harder to solve than (P_2). Another option would be to use the ellipsoidal trust region induced by linear models also for higher-order models.

5.3 Geometry improvements

Existing algorithms ensure that the optimization process will not get stuck because of a bad positioning of the design points on which the local models are based. Otherwise, the quality of the fitted models might become very poor and wrong conclusions are drawn. This problem is usually dealt with by incorporating a geometry check. When this check points out that the geometry of the design points is poor, a geometry improving evaluation is carried out instead of an objective improving evaluation. Different methods for achieving this have been proposed. Marazzi and Nocedal (2002), for example, add certain constraints to the objective improving step to ensure that the solution to the objective problem is also acceptable for the geometry of the design points. Powell (1994b) and Conn *et al.* (1997) concentrate on the determinant of the extended design matrix.

We describe how to incorporate the ideas behind the new ellipsoidal trust region also in the geometry improving iterations of the optimization process. We discuss the difference with the method used by Powell (1994b) and Conn *et al.* (1997). Finally, we show the correspondence between our geometry improving step and the D-optimality criterion in statistical DoE.

Powell (1994b) proposes to concentrate on the determinant of the extended design matrix, $\det(X)$. He uses interpolation to find local approximations and therefore the extended design matrix X is always square in his method. Conn *et al.* (1997) also apply this approach. The mathematical program that must be solved to find the geometry improving point is

$$\begin{aligned} &\max_{d} \ \det\left(X_{-i}(d)\right) \\ &\text{s.t.} \left\| \ d - d^* \ \right\|_2 \leq \Delta, \end{aligned} \tag{P_3}$$

where $X_{-i}(d)$ denotes the extended design matrix after inclusion of design point d and deletion of design point i. In general the design point i that, when replaced by the new point found with help of problem (P_2), allows for the biggest increase in the value of the determinant is removed from the set. Powell reasons that the determinant of a square matrix is a measure for the degree of singularity of this matrix. It is desirable to work

with a non-singular extended design matrix as it is used for solving a linear system of equations to create the interpolation models. Golub and Van Loan (1996) (p. 82) though, point out that matrices with a low absolute value for the determinant exist that are far from singular, as well as matrices with a high absolute value of the determinant that are almost singular. Hence, in certain situations the determinant of a square matrix is not a good measure for the degree of singularity of this matrix. Nevertheless, *maximizing* the determinant of $X(d)$ could result in an improved geometry of the design points. Because Powell uses interpolation, and we are looking at the more general case in which there may be more design points than parameters to be estimated, our matrix X is in general not square and we cannot use his measure for geometry improvement.

A first way to implement a geometry improving step is to use the D-optimality criterion. A second possibility is to use our trust region matrix C. The first approach is inspired by the commonly used D-optimality criterion from DoE (see Myers and Montgomery (1995)). In DoE the problem of extending an existing design in the best possible way is a well-known problem. By intuition, a geometry improving step is performed when the locations of the design points are such that some dimensions of the design space are hardly explored. By performing a geometry improving step we wish to maximize the amount of information that can be obtained from the design. This is exactly what the D-optimality criterion is about. A design of experiments is D-optimal when the generalized variance, $\det(X'X)^{-1}$, is minimized. This minimization is desired because the hyper volume of the joint confidence region of the $\beta's$ is proportional to $\sqrt{\det(X'X)^{-1}}$. Not only the volume, but also the ellipsoidal shape of the confidence region depends on $X'X$. Hence, the criterion based on D-optimality is to minimize $\det((X(d)'X(d))^{-1})$, i.e., to find the d that, when added to the extended design matrix X, leads to a minimal value for $\det(X'X)^{-1}$.

The second approach is based on our trust region matrix C. It is logical to aim at maximizing the volume of the trust region in (P_2) in a geometry improving step. For a fixed value of ρ maximizing the volume of our ellipsoid is equivalent with maximizing the determinant of the matrix C^{-1}, as the volume of the ellipsoid is given by

$$\mu_q \sqrt{\det(\rho^2 C^{-1})},$$

where μ_q denotes the volume of the q-dimensional unit sphere, which depends only on q. Hence, the second criterion is to maximize $\det(C(d)^{-1})$ or equivalently, to minimize $\det(C(d))$, i.e., to find the d such that, when added to the design matrix D, leads to a minimal value for $\det(C)$.

These two different approaches lead to two possible objectives in the geometry improving step, minimize $\det((X(d)'X(d))^{-1})$ and minimize $\det(C(d))$. In the following theorem we prove that these two objectives lead to the same optimal solution.

Theorem 5.2 *The matrix C, defined in (5.2), satisfies*

$$\det(C) = n \det(X'X)^{-1}.$$

Furthermore,

$$C^{-1} = D'D - n\bar{d}\bar{d}'.$$

Proof We recall that $X'X$ has a special structure (see (5.3)). The following relation holds for the matrix $X'X$ (Schur complement):

$$\begin{pmatrix} 1 & 0 \\ -\bar{d} & I \end{pmatrix} \begin{pmatrix} n & n\bar{d}' \\ n\bar{d} & D'D \end{pmatrix} = \begin{pmatrix} n & n\bar{d}' \\ 0 & D'D - n\bar{d}\bar{d}' \end{pmatrix}.$$

Hence

$$\det(X'X) = n \det(D'D - n\bar{d}\bar{d}')$$

and

$$\det(X'X)^{-1} = \frac{1}{n \det(D'D - n\bar{d}\bar{d}')}. \tag{5.6}$$

By substituting equations (5.2) and (5.3) into the identity

$$(X'X)^{-1}(X'X) = I$$

and then decompose it, we find that

$$\begin{pmatrix} na + nb'\bar{d} & na\bar{d}' + b'D'D \\ nb + nC\bar{d} & nb\bar{d}' + CD'D \end{pmatrix} = \begin{pmatrix} 1 & 0' \\ 0 & I_{q\times q} \end{pmatrix}. \tag{5.7}$$

By comparing the two block entries on position (2, 1) of this identity we find that

$$\bar{d} = -C^{-1}b, \tag{5.8}$$

From the block entries (2, 2) in (5.7) combined with (5.8) it follows that

$$C^{-1} = D'D - n\bar{d}\bar{d}'.$$

Hence

$$\det C = \frac{1}{\det(D'D - n\bar{d}\bar{d}')}. \tag{5.9}$$

The proposition follows by combining (5.6) and (5.9). □

We conclude that minimization of $\det(C)$ is a good geometry improving objective, both

from a theoretical as well as from an intuitive point of view: in DoE a lot of research has been done in D-optimal designs and they have proved to work well, and intuitively it is appealing to maximize the volume of the trust region. Note that for the special case of interpolation the geometry objective used by Powell (1994b) and the one we derived here, are equivalent. Hence, we provided different motivations for minimization of $\det(C)$ as geometry objective.

Besides the objective function we also have to constrain the area in which the best geometry improving design should be located. Without such a region constraint, the optimal design point would be located as far as possible away from the other design points. Of course, a design point too far away is not useful anymore for fitting *local* approximating models. Obviously, to use the ellipsoidal trust region here as well, is not applicable. A bad geometry means that there are some dimensions of the design space that are not enough explored compared to others. The shape of the ellipsoidal trust region reflects this bad positioning by a small range for the relatively unexplored dimensions and a large range for the other dimensions. Figure 5.2 illustrates the situation. This implies that, when using the ellipsoidal trust region, the search region for the geometry improving point is very narrow in the dimensions we are most interested in to explore more.

Another disadvantage of using the ellipsoidal trust region in the geometry improving step is the following. Dykstra (1971) showed that all the points on the boundary of our trust region constraint in (P_2) lead to the same value of the updated generalized variance, $\det(X(d)'X(d))^{-1}$. As our objective in the geometry improving step is to minimize this generalized variance, any point on the ellipsoidal trust region constraint would be optimal.

If the design problem is scaled in such a way that all dimensions are of equal magnitude, the classical spherical trust region would be most appropriate as trust region for the geometry improving step. Unlike in the objective improving step, where the trust region should reflect the fidelity of the models, in the geometry improving step the trust region should allow for dispersion of design points. In a well scaled design space a sphere stands for an optimal dispersion. Under the assumption of a proper scaling we propose to apply the following trust region for the geometry improving step

$$\left\| d - d^* \right\|_2 \leq \tau,$$

where τ denotes the radius of the area in which the optimal geometry improving point should be searched for and d^* denotes the best design found so far. As the area around this design d^* is most interesting, we wish to ensure a good geometry in that area.

The mathematical program that has to be solved to find the geometry improving point is

$$\begin{aligned} &\min_{d} \ \det C(d) \\ &\text{s.t.} \left\| d - d^* \right\|_2 \leq \tau. \end{aligned} \tag{P_4}$$

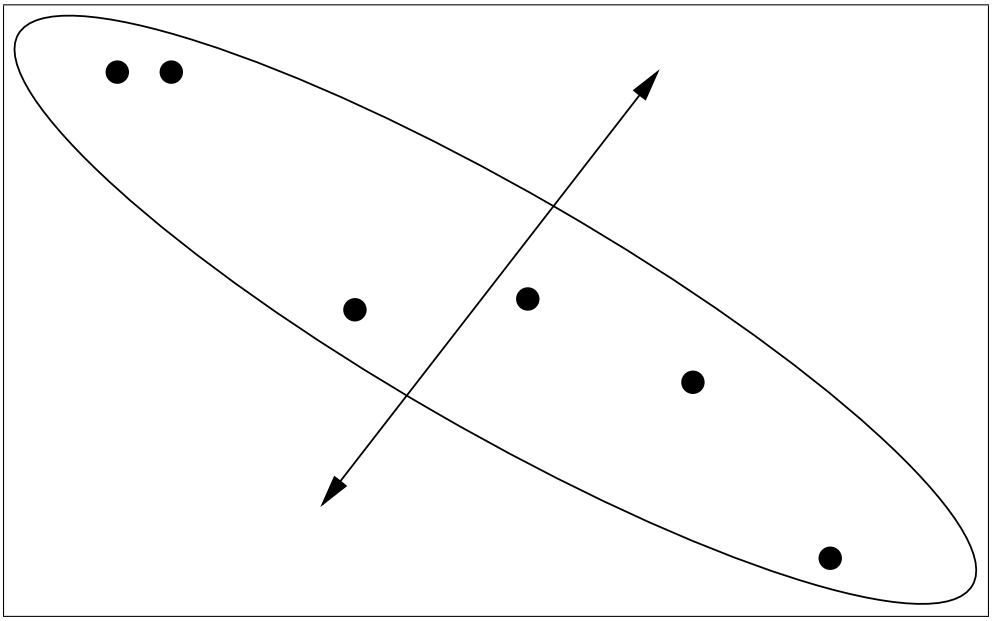

Figure 5.2: The direction in which exploration is most desirable for geometry improvement is also most restricted by the ellipsoidal trust region.

Dykstra (1971) has shown that the objective function in problem (P_4) is equivalent to a quadratic objective function. The problem then boils down to maximizing a non-concave quadratic form over a sphere. This particular problem is rather easy to solve and a global optimum can be found by using the solution method described in Ye (1992).

5.4 Properties of the ellipsoidal trust region

In this section we prove that use of our ellipsoidal trust region makes the objective improving step insensitive to affine transformations of the design space. First we show that use of the classical spherical trust region implies an objective improving step that is sensitive to such transformations.

The spherical trust region constraint

In the classical approach in each objective improving step problem (P_1) is solved. We show that the classical method including the spherical trust region is sensitive to affine transformations. Suppose that we transform the original problem by $\phi(d)$. Then we consider a method to be insensitive to affine transformations when the optimal solution of the original problem, d^*, and the optimal solution of the transformed problem, $\tilde{d}^*$ are related by $\tilde{d}^* = \phi(d^*)$. From now on, to distinguish between the variable space before and after the transformation we use the tilde sign above transformed variables.

Problem (P_1) is sensitive to a linear transformation of the variables d to $\tilde{d}$, defined by $\phi(d) = Md - s$, where M is a square, non-singular matrix of dimension $q \times q$ and s is a q-dimensional vector. In the rest of this paper M and s retain this meaning. If M is diagonal, pre-multiplication by it actually results in a scaling of the individual design parameters. Again we define $\tilde{x}' = [1\ \tilde{d}']$. Then it follows that the linear transformation ϕ results in $\tilde{x}' = [1\ (Md - s)']$. Hence, the constant term remains unaltered. Note that we can express the transformation ϕ in terms of x by pre-multiplying x by a matrix V of the following form

$$V = \begin{pmatrix} 1 & 0 \\ -s & M \end{pmatrix}, \tag{5.10}$$

We multiply the variables x with V. The extended design matrix $\tilde{X}$ then becomes

$$\tilde{X} = XV'.$$

Substituting this into the normal equations for $\tilde{\beta}_{+0}$, i.e., $\tilde{\beta}_{+0} = (\tilde{X}'\tilde{X})^{-1}\tilde{X}'y$, we find that[2]

$$\tilde{\beta}_{+0} = V^{-T}\beta_{+0},$$

from which follows that for the design space

$$\tilde{\beta} = M^{-T}\beta.$$

Hence, the transformed problem of (P_1)

$$\begin{aligned} &\max_{\tilde{d}}\ \tilde{\beta}'\tilde{d} \\ &\text{s.t. } \parallel \tilde{d} - \tilde{d}^* \parallel_2 \leq \Delta, \end{aligned}$$

can be rewritten to

$$\begin{aligned} &\max_{d}\ \beta' M^{-1} M d \\ &\text{s.t. } \parallel M(d - d^*) \parallel_2 \leq \Delta. \end{aligned}$$

which is equivalent to

$$\begin{aligned} &\max_{d}\ \beta' d \\ &\text{s.t. } \parallel M(d - d^*) \parallel_2 \leq \Delta. \end{aligned}$$

We see that *in general* the problem is *not* insensitive to a transformation with ϕ. The trust region constraint is transformed with M, while the objective did not change. This means that we search an optimal solution to the same objective function over a different

[2]For ease of exposition we use x^{-T} to denote the inverse transposed of x.

region now. We conclude that in general the objective improving step in the problem with a spherical trust region constraint is not insensitive to an affine transformation with ϕ. Note that this is caused by the fact that, though the objective is insensitive to such transformations, the trust region is not. We remark that for the special case of translation of d with a vector $s \in \mathbb{R}^q$, i.e. $M = I$, both d and d^* are translated with the same vector s and hence the feasible region does not change.

The ellipsoidal trust region constraint

In the following paragraph we take a closer look at the impact of using an ellipsoidal trust region in the objective improving step on the sensitivity to affine transformations. We show that our objective improving step, solving problem (P_2), is insensitive to affine transformations on the design space.

Theorem 5.3 *Problem (P_2) is insensitive to a linear transformation of the variables defined by $\phi(d) = Md - s$.*

Proof The only difference between problem (P_2) and the classical problem (P_1) is the fact that in the trust region constraint the C-norm is used instead of the 2-norm. We have shown already for problem (P_1) that the objective function is not influenced by a transformation by ϕ. We now take a closer look at the trust region constraint

$$|| \tilde{d} - \tilde{d}^* ||_{\tilde{C}} \leq \rho.$$

As $\tilde{d} = Md - s$, we can rewrite the left-hand side of this equation to

$$|| M(d - d^*) ||_{\tilde{C}} = (d - d^*)' M' \tilde{C} M (d - d^*).$$

Hence, if we can prove that $\tilde{C} = M^{-T} C M^{-1}$, then the trust region constraint is invariant under the transformation, which means that problem (P_2) is insensitive to the linear transformation ϕ.

We have seen already that the extended design matrix after transformation, $\tilde{X}$, equals XV'. Hence, the expression for $(\tilde{X}'\tilde{X})^{-1}$ becomes

$$(\tilde{X}'\tilde{X})^{-1} = V^{-T}(X'X)^{-1}V^{-1}. \tag{5.11}$$

With help of (5.2) and (5.10) we can rewrite (5.11) to

$$\begin{pmatrix} \tilde{a} & \tilde{b}' \\ \tilde{b} & \tilde{C} \end{pmatrix} = \begin{pmatrix} 1 & s'M^{-T} \\ 0 & M^{-T} \end{pmatrix} \begin{pmatrix} a & b' \\ b & C \end{pmatrix} \begin{pmatrix} 1 & 0 \\ M^{-1}s & M^{-1} \end{pmatrix}.$$

Writing out this block multiplication and concentrating on block element (2, 2) of the resulting right-hand side matrix, we find that

$$\tilde{C} = M^{-T} C M^{-1}.$$

□

5.5 Illustrative examples

In this section we discuss some preliminary numerical results of a comparison between using a spherical trust region and using our ellipsoidal trust region. First the tests that have been carried out are described. Then we give the test functions. Last, we present and discuss the results.

To be able to compare the ellipsoidal trust region with the spherical trust region independent of any sequential algorithm, we performed tests on a higher level. Starting point is a design space containing a set of design points, a test function f defined on this design space, and a spherical trust region around the design point with the best (minimal) observed objective value for f. We fit a linear regression model and solve the resulting optimization problem of minimizing the linear model subject to the spherical trust region constraint. The found optimum is stored for comparison with the ellipsoidal trust region approach, which is described next.

As explained in Section 5.2, the design points give rise to certain ellipsoids, the contour curves of the prediction variance. Each point in the spherical trust region is situated on one of these contour curves. We look for the point in the spherical trust region with the highest prediction variance, say V. As the spherical trust region allows for an uncertainty expressed by V, we allow the ellipsoidal trust region to contain points with at most the same uncertainty. Hence, the ellipsoidal trust region is situated around the point of gravity of the design points and has such a radius that design points on the border of the ellipsoid have prediction variance V, see Figure 5.3. We optimize the linear model over this ellipsoidal trust region and compare the found optimum with the one that arises from optimizing over the spherical trust region.

We performed tests with three different simple test functions, under different settings. For each test function the number of random design points on which the linear models will be based and the radius of the spherical trust region are varied. The test functions are given below. Function f_3 is Bigg's Function, problem no 267 in Schittkowski (1987):

$$
\begin{aligned}
&f_1(x_1, x_2) = 10x_1 + 3x_2,\\
&f_2(x_1, x_2) = x_1^2 + 3x_2^2,\\
&f_3(x_1, ..., x_5) = \sum_{j=1}^{11}(x_3 \exp(-\frac{jx_1}{10}) - x_4 \exp(-\frac{jx_2}{10}) + 3\exp(-\frac{jx_5}{10}) - y_j)^2,\\
&\text{where } y_j = \exp(-\frac{j}{10}) - 5\exp(-j) + 3\exp(-\frac{4j}{10}).
\end{aligned}
$$

Tables 5.1 and 5.2 summarize the different test settings together with the results. The first column specifies the test function. The second column denotes the number of design points used to build the linear model. We sampled the design points from a hypercube,

with radius equal to the trust region radius in column 3, centered at starting point (2.75, 2.75) for f_1 and f_2, and at starting point (2, 2, 2, 2, 2) for f_3. The optima are known to be (0, 0) for f_1 and f_2, and (1, 10, 1, 5, 4) for f_3. In Table 5.1 the influence of noise is investigated, by perturbing the function values by random noise. Column 4 of Table 5.1 specifies the standard deviation σ of the $N(0, \sigma)$ normal distribution. The last column of both tables shows the percentage of cases in which the ellipsoidal trust region resulted in a better new design point than the spherical trust region with respect to function value. For each setting 100 runs have been carried out. Figure 5.3 shows an instance of the resulting trust regions for f_1.

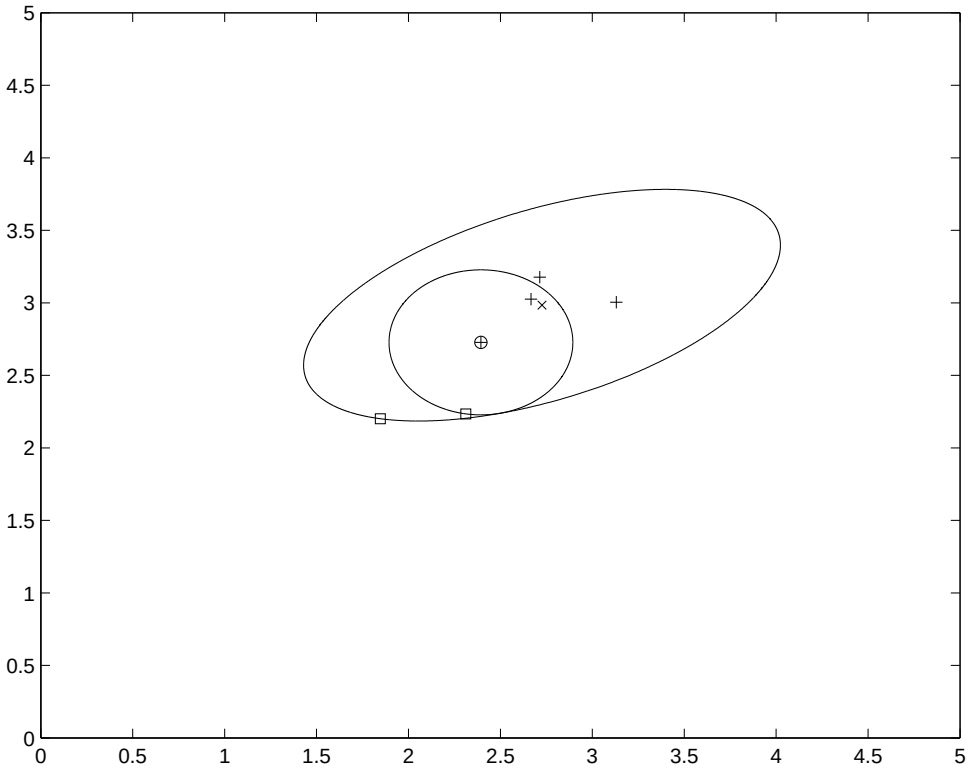

Figure 5.3: Comparison of ellipsoidal and spherical trust region for an instance of f_1. The '+' indicate the design points. The '□' indicate the optima found within the ellipsoidal and spherical trust region respectively. The 'x' denotes the center of the ellipsoidal trust region and the 'o' denotes the center of the spherical trust region.

These preliminary results show that the ellipsoidal trust region performs well for all settings for f_1 and f_2, both in the stochastic and in the deterministic setting. In case of the linear test function the linear approximating models exactly represent the test function. As the ellipsoidal trust region usually encloses the spherical trust region, a bigger gain is expected for the ellipsoidal trust region in this situation and the results are accordingly. For f_2 and f_3 the ellipsoidal trust region performs better for the smaller trust region. For all test functions a higher number of simulation points results in better performance of

function	nr points	trust region	σ	ellipsoid best
1	3	1	0.25	81
1	4	1	0.25	86
1	5	1	0.25	81
1	3	2	0.25	83
1	4	2	0.25	78
1	5	2	0.25	82
1	3	1	0.5	90
1	4	1	0.5	91
1	5	1	0.5	92
1	3	2	0.5	91
1	4	2	0.5	81
1	5	2	0.5	87
2	3	1	0.25	84
2	4	1	0.25	86
2	5	1	0.25	84
2	3	2	0.25	72
2	4	2	0.25	79
2	5	2	0.25	74
2	3	1	0.5	69
2	4	1	0.5	85
2	5	1	0.5	87
2	3	2	0.5	73
2	4	2	0.5	79
2	5	2	0.5	81
3	6	0.1	0.05	77
3	7	0.1	0.05	97
3	8	0.1	0.05	98
3	6	0.2	0.05	69
3	7	0.2	0.05	86
3	8	0.2	0.05	92
3	6	0.1	0.1	46
3	7	0.1	0.1	64
3	8	0.1	0.1	75
3	6	0.2	0.1	43
3	7	0.2	0.1	79
3	8	0.2	0.1	88

Table 5.1: Test results for noisy functions.

the ellipsoidal trust region. In the presence of noise the ellipsoidal trust region performs very well.

function	nr points	trust region	ellipsoid best
1	3	0.25	100
1	4	0.25	100
1	5	0.25	100
1	3	0.5	100
1	4	0.5	100
1	5	0.5	100
2	3	0.25	79
2	4	0.25	95
2	5	0.25	96
2	3	0.5	68
2	4	0.5	83
2	5	0.5	88
3	6	0.05	69
3	7	0.05	97
3	8	0.05	99
3	6	0.1	27
3	7	0.1	64
3	8	0.1	86

Table 5.2: Test results for deterministic functions.

5.6 Conclusions and future research

In this paper we proposed two new ideas that can be used in sequential derivative-free optimization methods. We discussed the use of an ellipsoidal trust region constraint based on statistical D-optimality in the objective improving step. The most attractive feature of this trust region is the fact that its shape and center are dependent on the location of the design points on which the approximating models are based. The D-optimality criterion is also used as geometry improvement objective. We showed the intuition behind the incorporation of the D-optimality criterion. Furthermore, we proved the independency of affine transformations for the ellipsoidal trust region.

The preliminary numerical results look promising. As further research it would be interesting to replace the spherical trust region with our ellipsoidal one in the existing sequential algorithms and to use the D-optimality criterion as geometry improvement objective.

Chapter 6

Why methods for optimization problems with time consuming function evaluations and integer variables should use global approximation models

Abstract: This paper advocates the use of methods based on *global* approximation models for optimization problems with time consuming function evaluations and integer variables. We show that methods based on local approximations may lead to the integer rounding of the optimal solution of the continuous problem, and even to worse solutions. Then we discuss a method based on global approximations. Test results show that such a method performs well, both for theoretical and practical examples, without suffering the disadvantages of methods based on local approximations.

6.1 Introduction

In this paper we consider the following integer optimization problem:

$$\begin{array}{ll} \min\limits_{x} & F(x, s(x)) \\ \text{s.t.} & x \in D, \end{array} \tag{6.1}$$

where $x \in \mathbb{Z}^p$ is a vector of integer design variables, $s(x) \in \mathbb{R}^k$ is a vector of simulation outcomes (also named responses), that depend on the value of x, F is an explicitly known, real-valued objective function, and $D \subset \mathbb{Z}^p$ is an explicitly known bounded region of feasible design points. The relationship between $s(x)$ and x is implicitly known and evaluating the value of $s(x)$ for a certain x is time consuming. This relationship could be defined in a black-box simulation model, for example. We assume that Problem (6.1) is deterministic and that the vector x consists of a limited number of integer or discrete

variables.

In engineering design problems integer or discrete variables often arise, for example the number of fins on a heatsink. Applications of the above optimization problem can also be found in logistic process design and analysis. In the last decade computer simulations are widely used in both fields. Simulation times are often high, sometimes even hours or a full day. Each scenario corresponds with a specific setting of the design variables. Due to time restrictions, only a limited number of scenarios can be evaluated and designers are confronted with the problem of deciding which scenarios to run. The crucial question becomes how to find the best possible design with a minimum number of simulations. There is no explicit information about the functional relationship between the design variables and the resulting process characteristics arising from the black-box computer model.

A broad range of distinct derivative free optimization approaches, mainly for continuous problems, can be distinguished. Schoofs *et al.* (1994), Toropov (1999), and Brekelmans *et al.* (2005) are examples of iterative optimization methods in which local polynomial approximations guide the search for an optimal design. Dennis and Torczon (1994), Torczon (1992), and Booker *et al.* (1999) describe (extensions of) a direct search method named pattern search. The pattern search framework consists of a search step and a polling step. The latter ensures convergence to a local optimum. The search step can involve any search strategy, from doing nothing at all to building global approximation models to speed up the search for an optimum. For an overview of derivative free optimization methods see Lewis *et al.* (2001).

Besides iterative algorithms based on local approximations, also non-iterative algorithms based on global approximations are proposed in the literature. See e.g. , Sacks *et al.* (1989), Balabanov *et al.* (1999), and Den Hertog and Stehouwer (2002a). In these methods explicit global approximations for the functional relationships between the design variables and the resulting responses are constructed using Design of (Computer) Experiments techniques to create a simulation scheme. The advantage of global approximation methods is that global insight is obtained in the behavior of these responses. Moreover, when the objective function F changes or the design area D decreases, no new simulations have to be carried out to find a new optimum, since the global approximations still can be used. The disadvantage of this approach, however, is that for high-dimensional design problems, the number of simulations required to obtain good global approximations becomes too high.

A good way to circumvent this high number of initial simulations is to construct the global approximations iteratively. Starting from a small set of initial simulations, global approximating models are created and refined in subsequent steps of the algorithm. Jones *et al.* (1998), Björkman and Holmström (2000), and Regis and Shoemaker (2005)

describe such methods. See also Jones (2001b) for a good review. The above described design optimization methods mainly focus on continuous design variables. Several papers describe optimization methods based on local approximations for solving Problem (1), in which the design variables are integers. See e.g. Bremicker *et al.* (1990), Loh and Papalambros (1991b), Loh and Papalambros (1991a), Abspoel *et al.* (2001), Gijsbers (2003), and Jacobs (2004). These local approximations, based on evaluations around the current iterate, are trusted only in a certain neighborhood of the current iterate. This neighborhood is often called the trust region. The next iterate is determined by substituting the local approximations and by optimizing over the trust region. For a formal description see Jacobs (2004). Disadvantage of these methods is that the use of local approximation models can easily result in ending up in the neighborhood of the continuous optimum (which is in general not the same as the integer optimum), or even worse.

Other approaches which are used especially in the field of discrete event simulation, are for example local search, simulated annealing, and tabu search (e.g. Glover *et al.* (1996)). These methods, however, assume that one simulation run can be performed quickly and require many runs.

We insist on using global instead of local approximation models for Problem (6.1), in which the design variables are integers, as the use of local approximation models can easily result in ending up in the neighborhood of the continuous optimum, or even worse. Another disadvantage of methods based on local approximations is that, even for problems with continuous variables, they converge to a local optimum. Methods based on global approximations have a better global overview of the responses over the whole design space and hence, a larger chance to find the global optimum.

In this paper we propose a solution method based on global approximation models to solve Problem (6.1). The method iteratively creates Kriging approximation models for the different responses. These models are based on the simulation data obtained so far. Using these models an explicit optimization problem is solved to obtain a promising design for the next simulation. The initial approximation models are based on a maximin Latin Hypercube Design (LHD) of computer experiments.

The remainder of this paper is organized as follows. In Section 6.2 we discuss the disadvantages of using local approximation models. Section 6.3 elaborates on our proposed solution approach. Numerical results are presented in Section 6.4. In this section we also compare to OptQuest (Glover *et al.* (1999)), a commercial optimization package often used in discrete event simulation. Section 6.5 concludes.

6.2 Drawbacks of methods using local approximations

Using optimization methods based on local approximations for solving Problem 6.1 has several disadvantages. This section will discuss them with the help of two examples.

For integer linear programming (ILP) problems it is well-known that solving the continuous version of an ILP and then rounding the solution to an integer solution, often yields a bad solution. Therefore, special algorithms are developed for ILP problems. We now want to argue that for the integer Problem (6.1) methods based on local approximations may lead to the integer rounding of the optimal solution of the continuous problem, and even to worse solutions.

Suppose that our original Problem (6.1) is the following ILP:

$$\begin{array}{rcl} \max\limits_{x_1,x_2} & (1+2\varepsilon)x_1 + x_2 & \\ \text{s.t. } x_1 & \geq & 0, \\ x_2 & \geq & 0, \\ (1+\varepsilon)x_1 + x_2 & \leq & 10, \\ x_1 & \in & \mathbb{Z}, \\ x_2 & \in & \mathbb{Z}, \end{array} \tag{6.2}$$

where $s(x) = (1+2\varepsilon)x_1 + x_2$, $F(x, s(x)) = s(x)$, and $\varepsilon > 0$ is a small constant. Moreover, suppose that we use local linear approximations, which have a perfect fit in this case, and that our trust region is the unit box $\|x - x^k\|_\infty \leq 1$, in which x^k is the current iterate. It is easy to see that e.g. starting in $(0,0)$, we will get the following iterate sequence: $(1,1)$, $(2,2)$, $(3,3)$, $(4,4)$, $(5,4)$, $(6,3)$, $(7,2)$, $(8,1)$, $(9,0)$. Hence, the final solution is indeed an integer rounding of the optimal solution $(\frac{10}{1+\varepsilon}, 0)$ of the continuous problem, whereas the integer optimal solution is $(0,10)$. In Figure 6.1 all solution paths are shown. It appears that all feasible starting solutions end up in the wrong solution $(\frac{10}{1+\varepsilon}, 0)$, except when we start in $(0,9)$, which is very close to the integer optimal solution.

We now reduce the trust region to the unit sphere $\|x - x^k\|_2 \leq 1$. In Figure 6.2 the solution paths are shown for this case. The algorithm can converge to many different solutions, which are even worse than the integer rounding of the optimal solution of the continuous problem, $(9,0)$.

One would expect that enlarging the trust region will decrease the number of iterations. That is not necessarily always the case. Suppose that the trust region is the following ellipsoid: $\frac{1}{4}(x_1 - x_1^k)^2 + (x_2 - x_2^k)^2 \leq 1$. Then starting in $(0,0)$ leads to the following solution path: $(0,0)$, $(2,0)$, $(4,0)$, $(6,0)$, $(8,0)$, $(9,0)$, containing five steps. When we use the larger trust region $|x_1 - x_1^k| \leq 2, |x_2 - x_2^k| \leq 1$, the solution path becomes $(0,0)$, $(2,1)$,

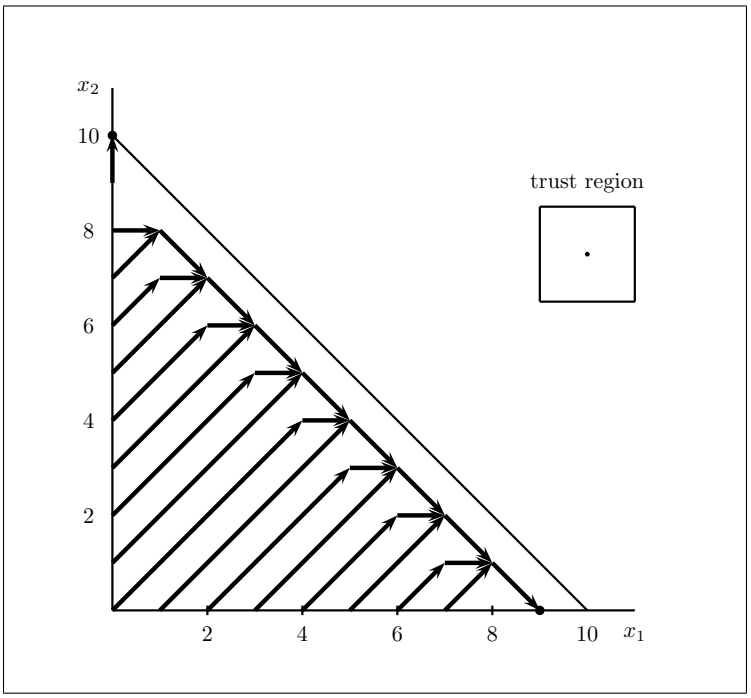

Figure 6.1: Solution paths for Problem (6.2) in case of a unit box as trust region.

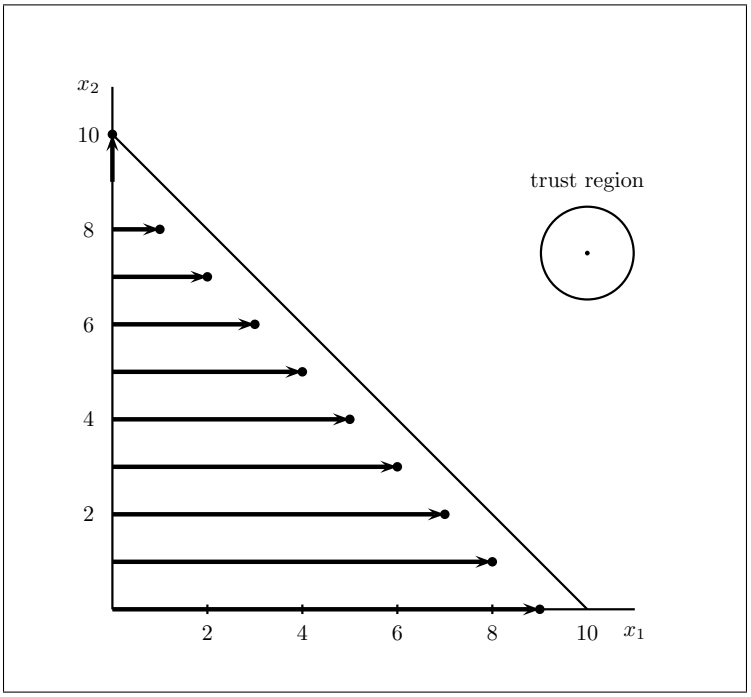

Figure 6.2: Solution paths for Problem (6.2) in case of a unit sphere as trust region.

$(4,2)$, $(6,3)$, $(7,2)$, $(8,1)$, $(9,0)$, which contains six steps.

Let us now look at the following slightly different problem:

$$\begin{array}{rcl} \max\limits_{x_1,x_2} & & (1+\varepsilon)x_1 + x_2 \\ \text{s.t. } x_1 & \geq & 0, \\ x_2 & \geq & 0, \\ (2+\varepsilon)x_1 + x_2 & \leq & 10, \\ x_1 & \in & \mathbb{Z}, \\ x_2 & \in & \mathbb{Z}, \end{array} \tag{6.3}$$

where $s(x) = (1+\varepsilon)x_1 + x_2$, $F(x, s(x)) = s(x)$, and $\varepsilon > 0$ is a small constant. Suppose the trust region is the unit box. Figure 6.3 shows the solution paths for this case. Again, the algorithm may end in different non-optimal solutions. Note that in this case the integer rounding of the continuous optimal solution is optimal, but only a few starting solutions close to the optimal solution, lead to this optimal solution.

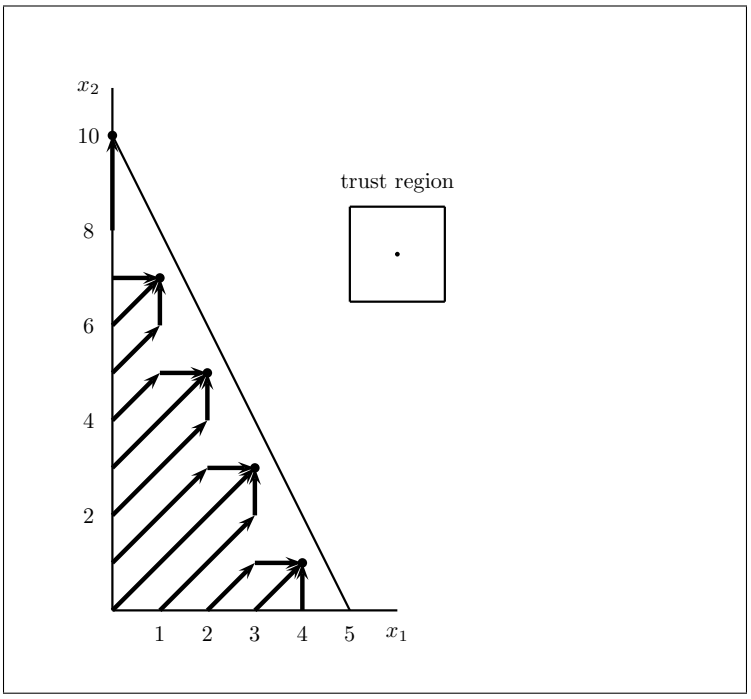

Figure 6.3: Solution paths for Problem (6.3) in case of a unit box as trust region.

Let us now enlarge the trust region as shown in Figure 6.4. This figure also shows the solutions paths. For several starting points the final solution is even worse than in Figure 6.3, which is based on a smaller trust region. For example, starting in $(0,0)$ leads to $(3,3)$ in Figure 6.3, and to the worse solution $(4,1)$ in Figure 6.4.

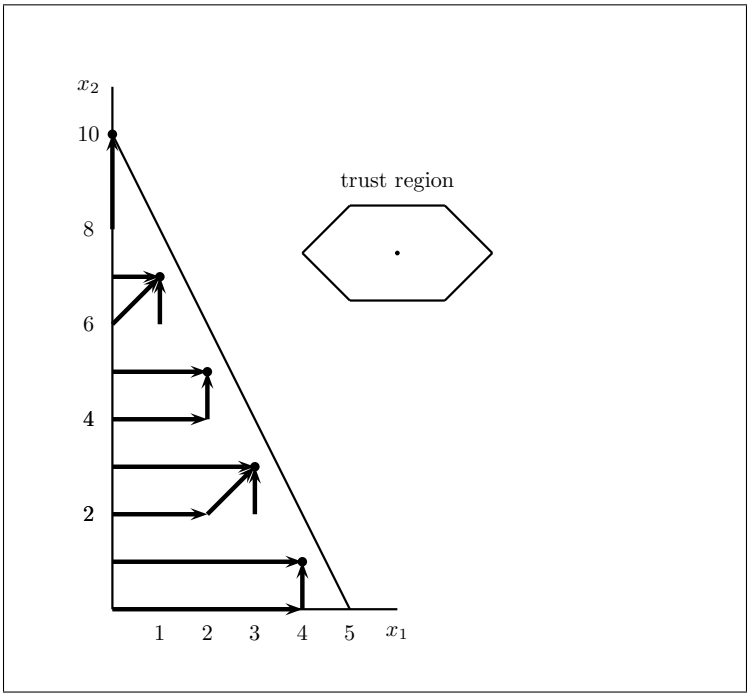

Figure 6.4: Solution paths for Problem (6.3) in case of an enlarged trust region.

Our final conclusion is that methods based on local approximations may end up in the wrong integer rounding of the continuous optimal solution. They may even end up in many different non-optimal solutions, depending on the starting solution. Moreover, enlarging the trust region may increase the number of iterations needed, and may lead to worse solutions. These conclusions are based on two-dimensional examples; for higher dimensional problems the effects can be even more disastrous.

We therefore plead for the use of global approximation methods in case of optimization problems when the number of variables is not too high. Building reliable global approximation models for a high-dimensional space requires a very large initial set of simulations. Hence, in case of many variables, global approximation models are out of reach and building local approximation models is the only alternative. In that case it is advisable to use a multi-start method to avoid getting stuck in local optima. In the next sections we propose and analyze an example of a method based on global approximations.

6.3 A method using global approximations

In the former section we discussed the disadvantages of local approximation methods for solving Problem (6.1). In this section we propose a method based on *global* approximation models. We will refer to this method as GAM (Global Approximation Method).

In this method the search for an optimal x is guided by the global approximation models $\hat{s}$. These models replace the unknown relationships s in Problem (6.1). Hence, in each iteration we solve

$$\begin{aligned} \min_{x} \quad & F(x, \hat{s}(x)) \\ \text{s.t.} \quad & x \in D \backslash Q, \end{aligned} \tag{6.4}$$

where Q indicates the set of already simulated designs.

Figure 6.5 presents a flowchart of GAM. The first step is to generate a Design of Computer Experiments of d points for the design variables x. For this initial set of design points, $\{x_1, ..., x_d\}$, simulations are carried out. This results in a set $P = \{(x_1, s(x_1)),$ ... , $(x_d, s(x_d))\}$ of design-response combinations. The vector s consists of k different simulation outcomes. For each element $s_i(x), i = 1, ..., k$, an approximation model $\hat{s}_i(x)$ can be constructed based on the data set P. Each $\hat{s}_i(x)$ approximates for any value $x \in D$ the corresponding value of $s_i(x)$. The approximation models are then used to locate the most promising value for x. Hence, as a next step we solve Problem (6.4), resulting in an optimum x^*. Global optimization solvers can be used to solve Problem (6.4), by defining the set of constraints needed to exclude Q from D. We simulate x^* to obtain $s(x^*)$. Note that this simulation is the most time consuming operation of the iteration. If the approximation models are adequate, the difference between the predicted values $\hat{s}(x^*)$ and the real values $s(x^*)$ is small. The quality of the approximation models in general improves as more iterations have been carried out. The next step is to check the stopping conditions. These stopping conditions could include a maximum number of simulations, a maximum amount of simulation time, and a satisfaction level with respect to objective function value, for example. If the stopping conditions are satisfied, we are done. Otherwise we add the new combination $(x^*, s(x^*))$ to the set P of points on which the approximation models are based and start the next iteration of constructing approximation models, etc. Having described the main flow of GAM, there are two elements that deserve some more attention: the initial set of simulations and the approximation models. They are discussed below.

Initial set of simulations

For building global approximation models at least a small initial set of simulations is needed. The optimal size of this set depends on the type of the approximation model, the number of input variables, and possibly on prior knowledge about the nonlinearity of the response to be approximated. The problem of choosing the initial set is called Design of Experiments (DOE) (Montgomery (1984)). Many different DOE schemes exist. Well known classical designs are the full / fractional factorial designs, Box-Behnken designs,

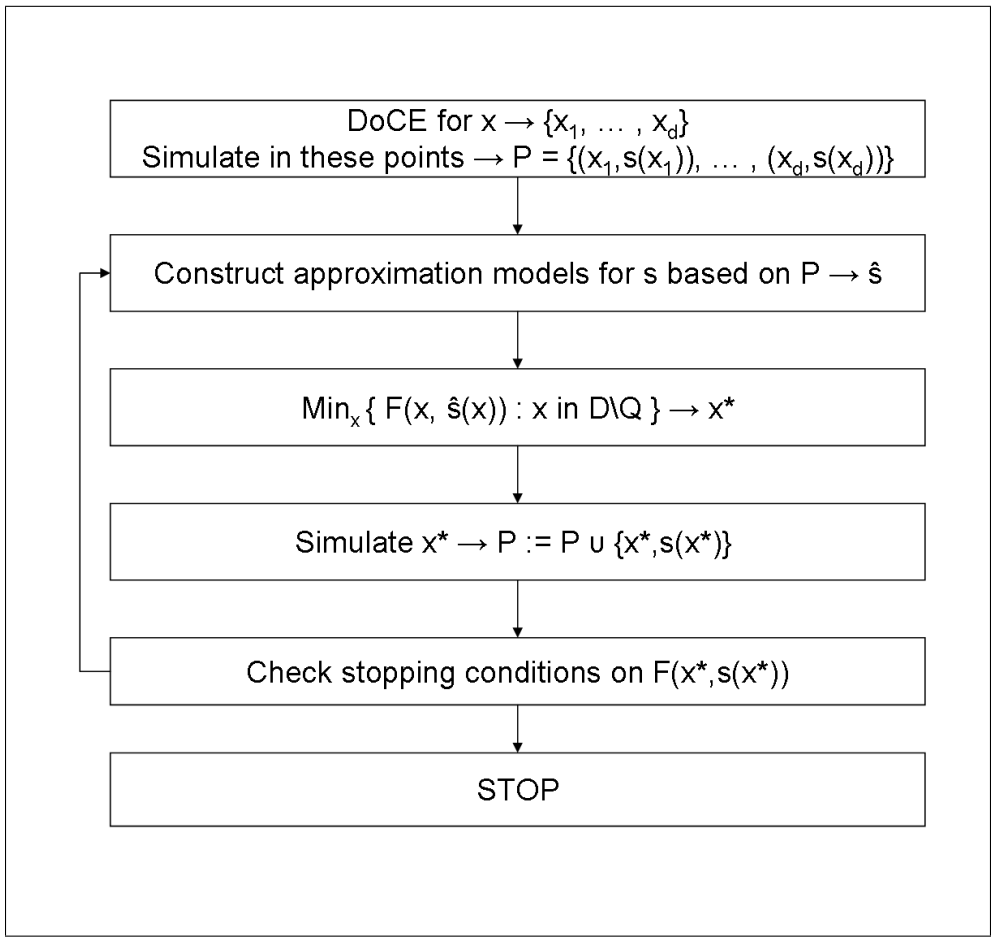

Figure 6.5: Flowchart of GAM.

central composite designs, and D-optimal designs. Another class of designs, Design of Computer Experiments (DoCE), has been specially developed for use with computer simulations (see Koehler and Owen (1996)). Examples are space-filling designs, random designs, orthogonal arrays, and LHDs.

In DoCE a desirable property of initial schemes is space-fillingness. One has to choose the points such that as much information as possible is captured. Intuitively this is the case when the points are spread throughout the feasible region as evenly as possible, i.e., the scheme is space-filling. Hereby we assume that no information is available about the function to be approximated. Another desirable property for a scheme is to be non-collapsing, which implies that for each input dimension separately the variable values are all distinct and well spread. Stehouwer and Den Hertog (1999) developed an approach for generating space-filling non-collapsing schemes for continuous variables. This method searches for the most space-filling scheme within the class of LHDs. It extends the approach presented by Morris and Mitchell (1995). The approach iterates over two steps. The first step constructs a random LHD. The second step searches for the best possible LHD with respect to space-fillingness. This search is performed using Simulated Annealing and the objective function guiding it is (Morris and Mitchell (1995))

$$\varphi_g(D) = (\sum_{j=1}^{m} J_j r_j^{-g})^{\frac{1}{g}}, \tag{6.5}$$

where g is a positive integer, r_j is the j^{th} element of a distance list $(r_1, ..., r_m)$ in which the elements are the distinct values of the inter-point distances, sorted from the smallest to the largest, and J_j is the number of pairs of points in D separated by distance r_j.

The algorithm of Stehouwer and Den Hertog (1999) has been extended to be able to deal with integer variables. In this case the LHD property is only valid when there are fewer initial points than integer levels in a certain dimension. The LHD property becomes a soft constraint for the integer variant: collapsing schemes get a high penalty in the objective function. The space-filling property remains valid in all cases. Figure 6.6 illustrates this.

Approximation models

The choice of the approximation model type is not an easy one. Depending on the nonlinearity of the response a more sophisticated approximation model is required. To be able to deal with nonlinear behavior as well we chose to use the Kriging model as approximation model, which was introduced as approximating functions for deterministic computer simulations by Sacks *et al.* (1989). In a Kriging model the response function $s_i(x)$ is treated as a realization of a stochastic process, just like in linear regression. A difference

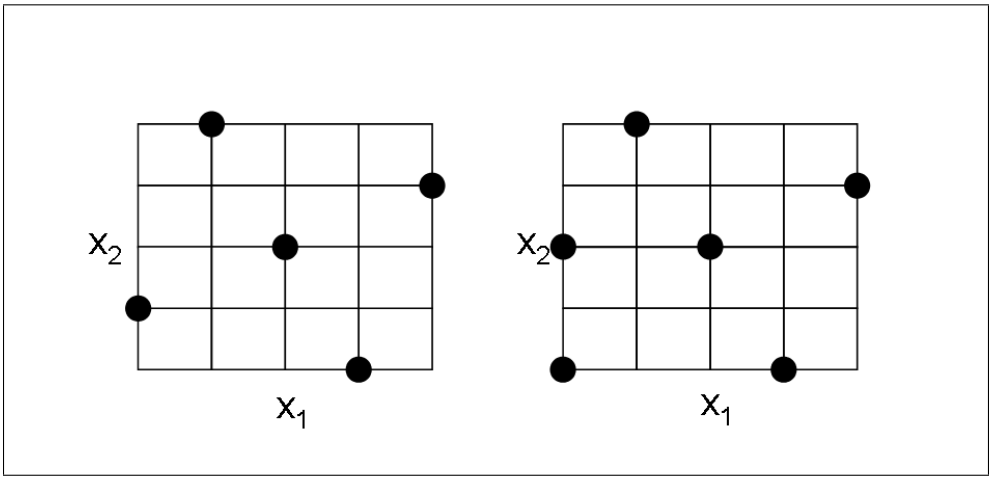

Figure 6.6: DoCE for a 2D design space. Both x_1 and x_2 have 5 possible integer levels. The picture to the left shows a DoCE with 5 design points, the picture to the right shows a DoCE with 6 design points. In the latter picture it is impossible to fully maintain the LHD property.

between a Kriging model and a regression model is that the Kriging model uses spatial correlation: the correlation between two data points is larger as they lie closer together. Another difference is that Kriging functions are interpolating. In fact the Kriging function is the Best Linear Unbiased Predictor. The resulting Kriging approximation function is of the following form

$$\hat{s}_i(x) = c_{i0} + \sum_{k=1}^{d} c_{ik} e^{-\sum_{j=1}^{p} \theta_{ij} \|x_j - x_j^k\|^{p_{ij}}}, \tag{6.6}$$

in which the model coefficients c_{i0}, c_{ik}, θ_{ij}, and p_{ij} depend on the set of data points on which the Kriging model is fitted. The coefficients θ_{ij} and p_{ij} are obtained by solving a Maximum Likelihood Estimator problem. Often the choice is made to keep $p_{ij} = 2$ fixed to obtain smooth Kriging models. Finding the coefficients of a Kriging model is computationally expensive, but much less expensive than a black-box simulation. We use a pattern search technique to find the coefficients θ_{ij} and keep $p_{ij} = 2$ fixed. The coefficients c_{i0} and c_{ik} follow from the BLUP property. More details about Kriging models, their underlying assumptions, and how to calculate the model coefficients can be found in Sacks *et al.* (1989).

For problems with a large variable range the described method is insufficient and requires extension. A well known issue for methods based on global approximations for continuous optimization problems, is the need for so called geometry improving iterations in the optimization process. Without such mechanism the optimization process can easily get stuck proposing only candidates that are in the neighborhood of the best solution so

far, as the global approximation model probably predicts a new optimum close to that best solution so far. To ensure global exploration, also proposals based on the location of a point in the design space instead of on its objective value are generated. Points located far away from already simulated points are most interesting in this respect as they apparently are located in relatively unexplored areas of the design space. Regis and Shoemaker (2005) describe an elegant way to ensure sufficient global exploration. They add an extra set of constraints that ensures that the next proposal is far enough away from already simulated points. The definition of what is far enough away is altered in each iteration and ranges from no restriction at all (local search) to requiring a large distance to already simulated points (global search).

For integer optimization problems with a large variable range, the above described need for geometry improvement arises, as these variables behave almost continuously.

6.4 Test results

In this section we discuss the results obtained on a set of academic test cases and compare these results with OptQuest, a commercial optimization package often used in discrete event simulation. Thereafter we present our results on two practical applications.

As a first test of the performance of GAM, we took the two linear problems described in Section 6.2. GAM found the integer optimum in 6 iterations for Problem (6.2) and in 7 iterations for Problem (6.3).

As a next test, we used the set of academic problems listed in Table 6.1. In each problem the vector of responses consists of one single response, and $F(x, s(x)) = s(x)$. To create a discrete version of the problems in Table 6.1, we introduced a stepsize and allowed for solutions ranging from lower to upper bound with steps of a size equal to the given stepsize.

Problem B is the Branin test function, Problem S is the Six-hump-camel-back test function, Problem R and Problem RL are instances of the generalized Rosenbrock function, and Problem L is the Levy function. The results on these test problems are listed in Table 6.2. We use the OptQuest Solver Free Demo Version 1.5 (OptQuest Version 4.0.5.2), with settings auto stop = off, database size = 10000, objective precision = 0.000001 and variables precision = 0.000001. We run the solver for different values of the number of iterations. Note that this demo version can only locate one optimum, hence for Problem S OptQuest finds only one of the two optima. In our implementation of GAM we use $g = 3$ in (6.5). Instead of using an NLP solver for Problem (6.4), we iterate through all feasible solutions and choose the best one.

We can conclude from the results in Table 6.2 that GAM finds the global optimum quickly and performs well in comparison to OptQuest. Moreover, Figure 6.7 shows that

	$s(x)$	p	Bounds	S.s.	# Scen.	Opt. F
B	$[x_2 - 5.1(\frac{x_1}{2\pi})^2 + \frac{5x_1}{\pi} - 6]^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	2	x_1:[2 10] x_2:[−2 4]	0.1	4941	0.403
S	$4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	x_1:[−1.5 1.5] x_2:[−1 1]	0.1	651	-1.030
Q	$x_1^2 + (x_2 - 2)^2 + 2x_3$	3	$x_{1..3}$:[−10 10]	1	9261	-20.000
R	$\sum_{i=1}^{4} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	5	$x_{1..5}$:[−5 5]	1	161051	0.000
L	$\sin^2 3\pi x_1 + (x_6 - 1)(1 + \sin^2 2\pi x_6) + \sum_{i=1}^{5}((x_i - 1)^2(1 + \sin^2 3\pi x_{i+1}))$	6	$x_{1..5}$:[−4 4] x_6:[−5 3]	1	532441	-6.000
X	$\sum_{i=1}^{7}(x_i^2 - 2\sin(2x_i))$	7	$x_{1..7}$:[−3 3]	1	823543	-5.730
RL	$\sum_{i=1}^{5} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	6	$x_{1..6}$:[−5 5]	1	1771561	0.000

Table 6.1: Problem characteristics of academic test problems. Column S.s. denotes the stepsize, and column # Scen. denotes the number of possible scenario's.

	Total # simulations GAM	Total # simulations OptQuest
B	8	674
S	11, 34 (*)	55
Q	14	311
R	31	290
L	20	351
X	65	38
RL	67	144

Table 6.2: Results on academic test problems.(*) This function has two global optima.

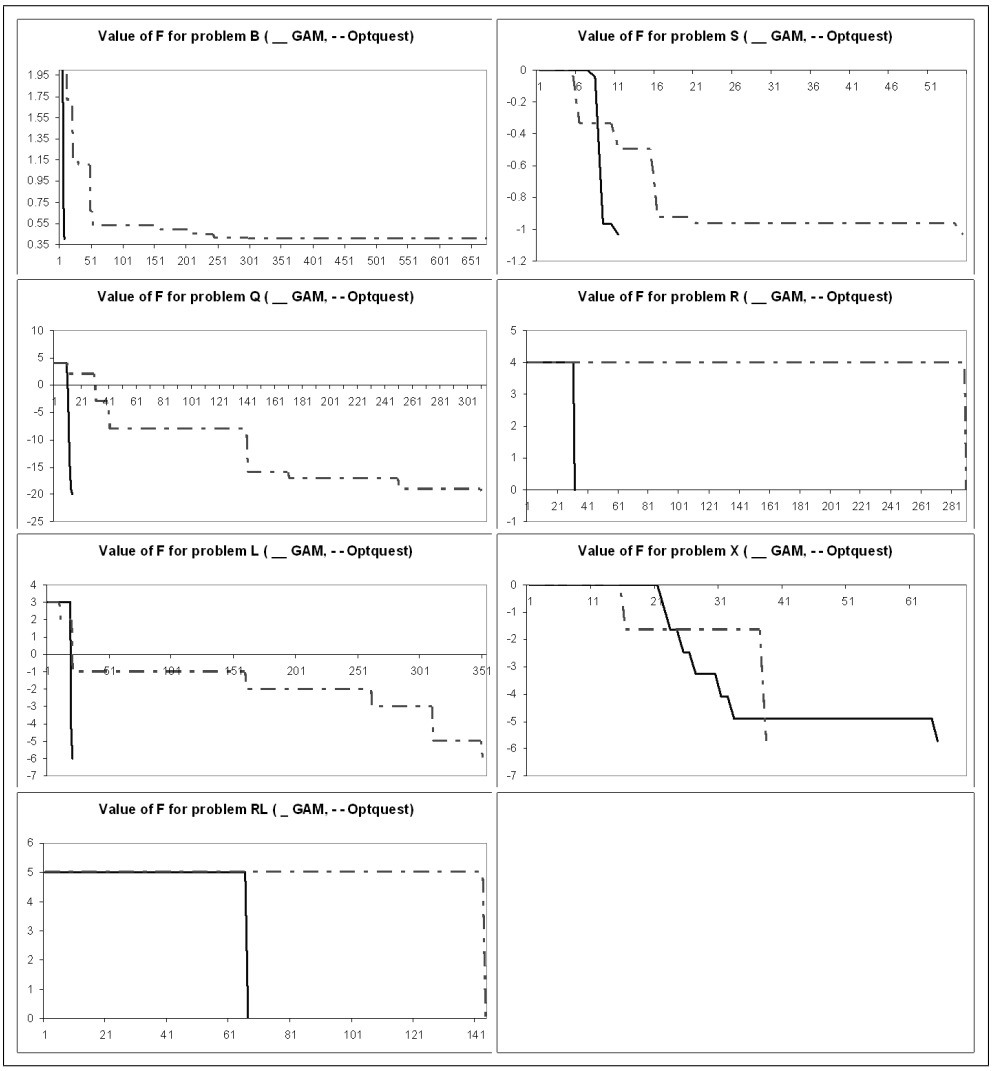

Figure 6.7: Subsequent iterations towards optimum for GAM and OptQuest.

GAM often finds large improvements almost immediately after the DoCE phase has been completed. In a situation in which each simulation takes hours and there is a restriction on the number of simulations that can be carried out, it is very important to find large improvements as quickly as possible. Taking problem L as example and assuming a simulation time of 1 hour, the total time involved with GAM is 191 seconds GAM calculation time plus 20 hours simulation time, whereas OptQuest would need 351 hours of simulation time. The larger the simulation time, the larger the difference in solution time becomes. Problem X is the only problem for which Optquest performs better.

Next we consider two practical applications, a location problem and a closed queuing network problem. The location problem is the problem of deciding which subset of the set of possible distribution centers should be opened and how the client locations should be assigned to the open distribution centers, such that the total of setup cost and transport cost is minimized. In the context of black-box optimization we regard the choice of which distribution centers should be opened as the black-box optimization problem. Hence, with C distribution centers there are 2^C possible solutions. The calculation of total costs, including the optimal assignment of client locations to distribution centers that are open, is carried out in a black-box simulation model. In Problem LocS, the small location problem, there are 5 available distribution centers. In Problem Loc, the larger location problem, there are 10 available distribution centers.

Closed queuing networks arise for example in manufacturing units attempting to maintain a constant level of work in process, or busy computer networks for which it is assumed that a new job enters the network as soon as another job leaves. Such systems, in which a constant number of jobs is present, may be modelled as closed queuing networks. In the small queuing problem we consider a flexible manufacturing system in which at all times 10 parts are present. There are 3 machines. Each part begins by having operation 1 done at machine 1. Then, with probability 0.75 the part has operation 2 processed on machine 2, and with probability 0.25 the part has operation 2 processed on machine 3. Once a part completes operation 2, it leaves the system and is immediately replaced by another part. Each machine can be purchased at 5 different machine rates. The fastest rate costs most. The optimization problem then becomes to choose the machine rates resulting in the highest number of service completions (in parts per minute) at lowest cost. Given the machine rates, a black-box simulation model is used to calculate the total cost (in terms of machine purchase prices and service completions). There are 125 different scenario's. In the larger queuing network there are 15 parts always present in the system and 5 machines that can be purchased at 5 different speeds, resulting in 3125 scenario's.

Problem characteristics and results for the location and queuing network problems are listed in Table 6.3 and Table 6.4. For all four problems GAM finds the optimum in very few iterations.

	$s(x)$	p	Bounds	S.s.	# Scenario's	Optimal F
LocS	small location	5	$x_{1..5}$:[0 1]	1	32	449215.3
Loc	location	10	$x_{1..10}$:[0 1]	1	1024	471500
QueuingS	small queuing	3	$x_{1..3}$:[−2 2]	1	125	-20001
Queuing	queuing	5	$x_{1..5}$:[−2 2]	1	3125	-0.804162

Table 6.3: Problem characteristics of practical applications.

	Total # simulations GAM
LocS	6
Loc	27
QueuingS	6
Queuing	19

Table 6.4: Results on practical applications.

The above results show that GAM, a rather basic method based on global approximation methods, is very effective for integer problems with a discrete nature. Applying GAM to integer problems with a large variable range for the design parameters we find the behavior we described at the end of Section 6.3: GAM finds a reasonably good point rather quickly and then lingers around in the neighborhood of this point. For these kind of problems GAM should be extended with a geometry safeguard mechanism, as described in Section 6.3.

6.5 Conclusion

We have shown that in case of integer design variables, optimization methods based on local approximation models risk to end up in the neighborhood of the continuous optimum or even worse, instead of in the integer optimum. Therefore, we believe that it is better to use global approximation models instead for integer design problems. We have shown that an optimization method based on global Kriging approximation models performs very well on a set of test problems with integer design variables. In case of expensive or time consuming simulations this method outperforms the optimization method OptQuest.

Chapter 7

Simulation-based design optimization methodologies applied to CFD

Abstract: Finding the optimal physical design for an electronic system is extremely time consuming. In this paper we describe a sequential global optimization methodology that can lead to better designs in less time, and illustrate its use by optimizing the design of a heat sink for a simple system. The results show the need for a global approach, the insights that can be gained through automated design optimization, and illustrate the efficiency of the reported methodology in finding the optimum design.

7.1 Introduction

As electronic products become more sophisticated and design margins tighten, defining the thermal management strategy early in the design cycle is vital to ensure a cost-effective design for the level of heat dissipation, and high field reliability. Optimizing the cooling system for electronic products can involve many design variables, such as airflow rate, fan and vent locations, heat sink size, etc.

Numerical tools, for example computational fluid dynamics (CFD) tools, are increasingly used in the physical design of electronic products to qualify and improve the design and reduce time to market. However, despite the computing power available, exploring all possible design alternatives is extremely time consuming.

Much research has already been carried out in the field of design optimization and numerous methods exist. Methods that explicitly take into account the high cost involved with a function evaluation can roughly be divided into two groups, sequential methods and non-sequential methods. Non-sequential methods are aimed at modeling the whole design space with help of dedicated Design of Experiments techniques and Response Surface Models. See for example Den Hertog and Stehouwer (2002a) and Den Hertog and Stehouwer (2002b). Use of a sequential method is more suitable for multi-dimensional design spaces containing infeasible areas. References for some sequential optimization

methods are Brekelmans *et al.* (2001), Marazzi and Nocedal (2002), Conn *et al.* (1997), and Toropov (1999).

The paper discusses a simulation-based sequential global optimization methodology developed for use with a Computational Fluid Dynamics (CFD) tool, FLOTHERM, and illustrates the application of the technology by optimizing the design of a plate fin heat sink. The main benefits of using automated design optimization are that better designs can be obtained in less time, and insights are gained into the performance of the design.

7.2 Overview of the optimization problem

The general picture is given in Figure 7.1. On the left is a set of Design Parameters that have to be set optimally. Typical examples are component locations, number of fins on a heat sink, etc. On the right are the relevant quantities calculated by the analysis tool, called Response Parameters. Examples are junction temperatures, pressure drops, fan flow rates, etc. The design optimization problem is to find a set of values for the design parameters such that the design parameters satisfy certain constraints, the response parameters satisfy certain constraints, and some Objective, being a function of the response parameters, is optimized.

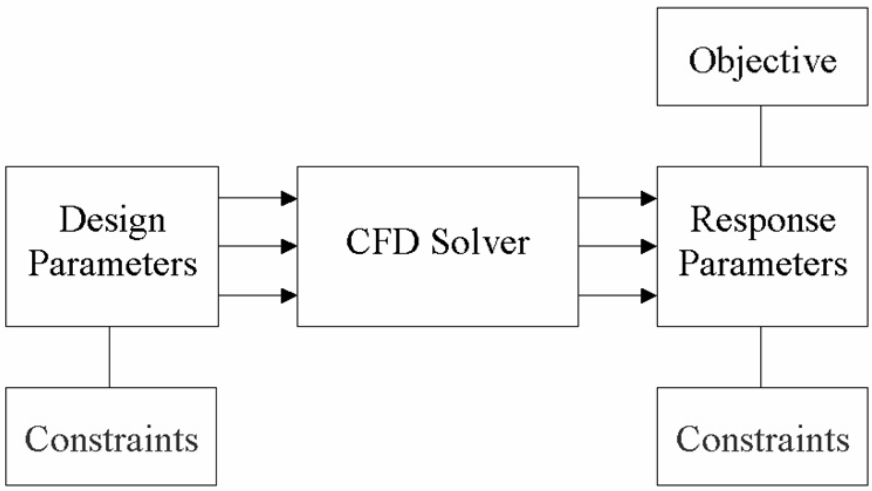

Figure 7.1: The design optimization problem.

The principal characteristics that make this class of problem quite complex are:

- Function evaluations (i.e. CFD runs) are time consuming
- Nonlinear objective function and nonlinear constraints

- The existence of local minima
- Numerical noise introduced by mesh changes
- Small residual errors in the solution due to the use of finite convergence criteria
- Absence of derivative (i.e. gradient) information
- Presence of integer design parameters like the number of fins on a heat sink
- Collision constraints (i.e. objects are prevented from colliding with other objects in the design space).

The optimization methodology described here deals with the complexities listed above. Compared to pre-existing sequential algorithms the novel aspects are:

- The global optimization approach
- Dealing with integer design parameters, and
- Dealing with object collision constraints.

The approach consists of two steps: an explorative search of the design space, followed by a local optimization. These steps are outlined next.

7.3 Explorative search (ES)

The possible existence of local minima makes it necessary to apply a global optimization strategy. First we generate an initial set of designs or Design Points within the feasible design space that will be run by the CFD tool. The purpose of this set of design points is twofold:

- Most importantly, the design space is explored to locate interesting areas to be further explored in the local optimization step.
- It provides a set of base points or support vectors that are subsequently exploited during the execution of the sequential optimization algorithm.

To create a set of well chosen starting points, a Latin Hypercube Design (LHD) is generated. Stehouwer and Den Hertog (1999) describe the construction of such LHDs for continuous valued design parameters with constraints. These LHDs possess the following characteristics, which are important for simulation-based experiments:

- Space Filling – designs are spread throughout the design space as evenly as possible

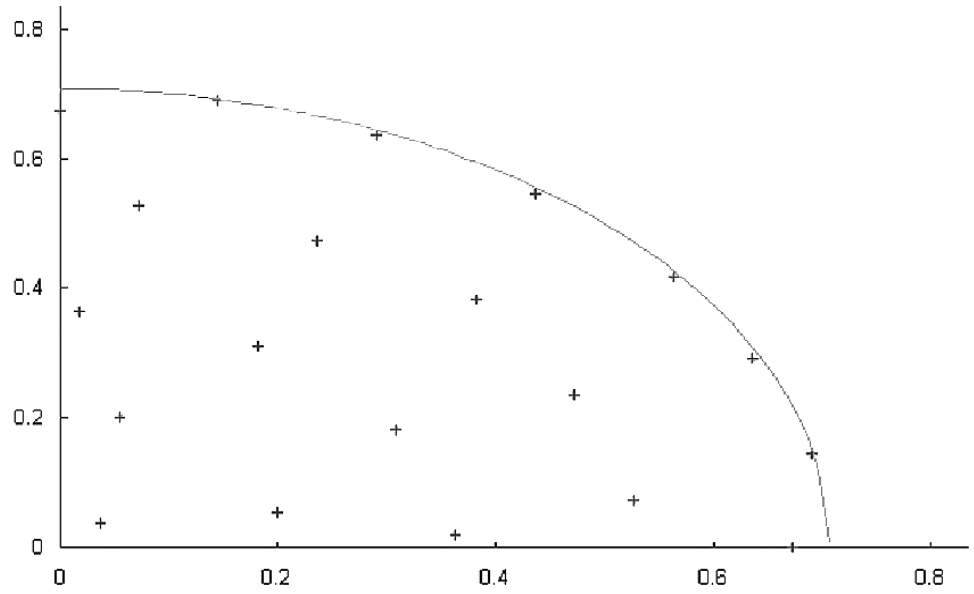

Figure 7.2: A 2D space-filling LHD.

- Non-Collapsing – each design parameter has a unique value, i.e. different for every design point
- Non-box – able to handle nonlinearly constrained design spaces with infeasible regions

We do not propagate the use of classical simulation schemes for computer experimentation for the following reasons. In computer experimentation noise does not play a role, since running a computer simulation twice generally yields exactly the same results, this in contrast with physical experimentation. In computer experimentation no information is gained from the repeated simulation of the same design point as is often done in classical DoE. Also due to the presence of noise, in physical experimentation it is often optimal to have design points lie on the borders of the design region. In computer experimentation other parts of the design region are often equally interesting. A last drawback of most classical experimental design methods is that they are only applicable for rectangular design regions. Figure 7.2 gives an example of the design points generated using a space-filling non-collapsing LHD for a nonlinearly constrained two-dimensional design space.

7.4 Local optimization (LO)

From a given starting point a so-called local optimization approach is used to converge to the nearest local optimum. Traditional optimization methods based on derivatives fail,

because derivative information is not available, and is too expensive to estimate through finite-differencing. Moreover, these methods may get trapped in non-physical local minima introduced by numerical noise. For example, the change in the result obtained when an object is moved slightly may be due more to changes in the grid distribution than to any physical effect. Hence a sequential optimization approach to deal with time consuming CFD runs without reliance on derivative information had to be developed, as briefly described below. Iteratively, the following steps are executed:

Step 1 Local linear approximations of the CFD model outputs are obtained with the help of weighted regression techniques on a [0, 1] scaled design and response domain. We ensure that the approximation is exact in the current best design. The weight of observation , i, w_i, depends on the distance between i and the current best design. Designs far away from the current best design receive a weight that is lower than the weight of designs that are close to the current best design. The formula we used is $w_i = e^{-p\|d_i - d^*\|^2}$, where d_i is design point i, d^*, is the current best design and $p = 20$. It can happen that the design points receiving a positive weight in the weighted regression do not form a basis for the n-dimensional design space. In that case no approximating models are generated and a geometry step is carried out instead.

Step 2 The resulting approximate (or compact) models are then optimized within a T*rust Region* centered on the current best design to find the best feasible objective improving point (i.e. a step towards the optimum).

Step 3 If the geometry of the design points that determine the local approximations are located in such a way that they result in a bad approximation of the actual model, then we evaluate a geometry improving point (to improve the accuracy of the approximate model) instead of an objective improving point. The geometry measure is based on the Euclidian distance to the closest other design in the trust region for each design within this trust region. The geometry measure is the maximum over those minimal distances. We calculate this geometry measure twice, once for the current set of design points inside the trust region including the objective improving point and once for the current set of design points inside the trust region including the geometry improving point. The latter is determined by using the LHD routine to add one new design point to the existing set of points inside the trust region. If the geometry measure is much higher for the geometry improving point than for the objective improving point, the geometry improving point is located much further away from the set of design points within the trust region than the objective improving point and hence the geometry of the design points can be improved by

simulating the geometry improving point.

Step 4 At each iteration, a new local linear approximation is built, and either a new design point is evaluated (objective or geometry improving), or the trust region is decreased, depending on the progress of the optimizer.

The focus of the approach is on getting good solutions with a limited number of function evaluations. The termination criteria are, among others, the size of this trust region and a maximum number of simulation steps to be performed. For a more thorough description of a similar approach we refer to Brekelmans *et al.* (2001).

To aid the search for the global optimum, a multi-start strategy can be used, meaning that several local optimizations are started sequentially. The starting points generated in the explorative search step are visited in order of diminishing objective value. Given the computational expense of the CFD runs, there is a trade-off between precisely converging on the local optimum found from one starting point and moving to the next starting point in the hope of finding a better optimum.

7.5 Illustrative example

The example considered here illustrates the use of the optimization technology with CFD to help optimize the design of a heat sink. The example is deliberately simple to aid understanding, being a simple channel. The problem definition is as follows:

A 20mm x 20mm thermally enhanced board-mounted component, powered at 10W, has to be cooled below the design limit for the junction temperature of 95 ^{o}C (100 ^{o}C less a 5^{o}C safety margin), based on a local ambient temperature of 45^{o}C. The objective is to find the cheapest heat sink that is required to achieve this by natural convection for the given system configuration.

The absolute size constraints on the heat sink size are imposed by the system. Whereas the system design itself imposes no restriction on the number of fins, their thickness, the type of heat sink, nor it's fabrication method, choices based on experience can help ensure the optimization results in a practical design. a choice of the latter (e.g. extruded), sensible design parameter ranges can be set based on experience. If the optimum design is found to be at the limit of the range set, these can be considered further.

In this example, both the base thickness and the fin height are considered as design parameters. However, to ensure that the heat sink fits into the system the overall height of the heat sink is restricted by providing a constraint on these design parameters:

$$Base\ Thickness + Fin\ Height \leq 55mm$$

Design Parameter	Min value	Max value
Overall Height (Base + Fins)	6mm	55mm
Base Width	20mm	60mm
Base Length	20mm	60mm
Number of Fin	5	30
Fin Thickness	0.5mm	2.5mm
Base Thickness	1.0mm	5.0mm

Table 7.1: Design parameter ranges.

Response Parameter	Min value	Max value
Heat Sink Mass	N/A	N/A
Component Temp. Rise	N/A	50^oC

Table 7.2: Response Parameter Ranges.

This results in a design space that contains an infeasible region. However, this does not pose a problem for the approach global optimization approach reported here.

The optimization task presented here is challenging for several reasons:

- One of the parameters (number of fins) is an integer and therefore discontinuous
- The natural convection environment provides strong interaction between the heat sink geometry and the flow (radiation is ignored in this example)
- The range over which the parameters are being varied is quite large so the design points are widely spaced

The location of the heat sink is set as a function of the base width and base length being varied by the optimizer, such that the heat sink base remains centered on the component as its size changes, i.e.:

$$Base\ X\ Location = Constant - 0.5 * Base\ Length$$

$$Base\ Y\ Location = Constant - 0.5 * Base\ Width$$

We cannot directly minimize cost. However, for a given fabrication process and attachment method, this correlates with the mass of the heat sink, so the objective is chosen to be:

$$Objective = min(Heat\ Sink\ Mass)$$

The number of cells between the fins is held constant at 3, previously found to give good results for laminar flow between the fins Dyson (1993). However, the main reason

Overall Height	35.0mm
Base Width	60.0mm
Base Length	60.0mm
Number of Fins	12
Fin Thickness	0.50mm
Base Thickness	5.0mm
Component Temp. Rise	33.0^{o}C
Heat Sink Mass	80.6g

Table 7.3: Base case design.

Overall Height	25.3mm
Base Width	52.7mm
Base Length	52.7mm
Number of Fins	10
Fin Thickness	0.50mm
Base Thickness	4.3mm
Component Temp. Rise	44.9^{o}C
Heat Sink Mass	49.1g

Table 7.4: LO from base case design.

for keeping this constant for the study is to reduce the computational cost. Once a near optimal design has been found the influence of the mesh between the fins can be investigated by considering a narrower range for the other design parameters.

7.6 Results

For this example we have performed the following optimization studies:

- local optimization from the base case (LO);
- local optimization from the best design obtained from an initial space-filling set of approx. 30 experiments (ES+LO); and
- global optimization, by starting a local optimization from each design in the ES (ES+GO).

The number of designs required in the ES depends on the number of design variables and how nonlinear the resulting response surface is, which is generally not known a priori. As a rule of thumb, the number of designs is taken equal recommended to be 5 times the number of design variables, giving 30 runs in this case. It is important to emphasize that the optimization methodology being reported on here, is designed to achieve a substantial improvement in the objective within a limited simulation budget, dictated by time and computer resources, not to rigorously find the optimal solution. The two step approach employed offers considerable flexibility in this regard, as the three studies described above illustrate.

The design parameters for the base case, and the best feasible design from each of these three optimizations are shown in Table 7.3 to 7.6 below, together with the resulting response parameters values.

For each optimization the performance of the optimizer is shown in the graphs below. These show the heat sink mass for each step taken during the local optimization (SO step) and the resulting component temperature rise. Also reported is the mass of the current

Overall Height	35.2mm
Base Width	52.7mm
Base Length	20.0mm
Number of Fins	10
Fin Thickness	0.50mm
Base Thickness	2.0mm
Component Temp. Rise	49.6^{o}C
Heat Sink Mass	15.3g

Table 7.5: LO from best design in the ES.

Overall Height	42.9mm
Base Width	40.7mm
Base Length	26.7mm
Number of Fins	7
Fin Thickness	0.54mm
Base Thickness	1.0mm
Component Temp. Rise	49.9^{o}C
Heat Sink Mass	15.0g

Table 7.6: GO from all designs in the ES.

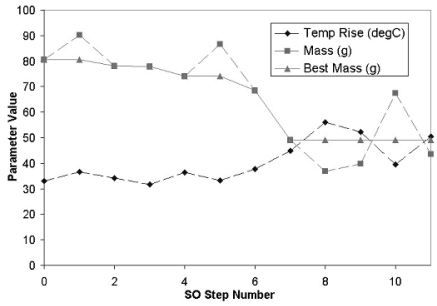

Figure 7.3: LO from base case.

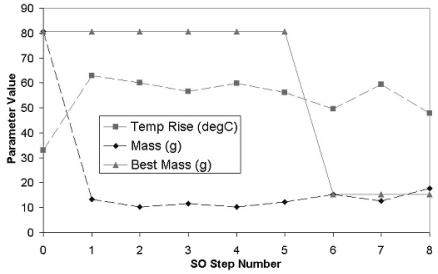

Figure 7.4: LO from best design in the ES.

best feasible design (best mass). This is the lowest mass heat sink satisfying the constraint on the component temperature rise achieved up to this point in the optimization process. In each graph the base case result is shown for SO Step 0 to provide a feasible initial design.

7.7 Discussion of results

In Figure 7.3, SO Step 1 gives a higher mass heat sink. The current best design is the original, and so the 'best mass' remains unchanged. Subsequent steps find improved designs until the local optimum is found at SO Step 7. The termination criteria for the search algorithm are set to ensure that computational effort is not wasted by finding the local optimum with unnecessarily high precision, given that the global optimum may well be in some other part of the design space. However, this is not a limitation of the algorithm.

In Figure 7.4, SO Steps 1 to 5 all give a lower heat sink mass. However, the temperature rise for each of these steps is too high, so SO Step 0 remains the current best design until SO Step 6 where the local optimum is found. This design, given in Table 7.5 is a quite different design from that found when starting from the base case, given in Table 7.4, providing evidence for the presence of multiple optima within the design space, and hence

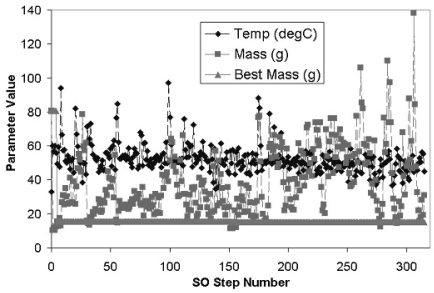

Figure 7.5: LO from every design in the ES.

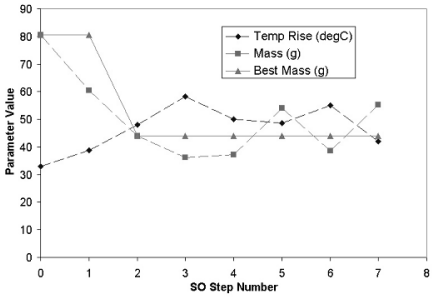

Figure 7.6: LO from base case during GO.

the need to use a global optimization approach.

The first few SO Steps in Figure 7.5 replicate those in Figure 7.4, since the GO starts with a LO from the best design in the ES before moving on to the next best design, and so on. This remains the optimum design until SO Step 152 where a final small improvement in the design is achieved. The LO performed from the base case as part of the GO follows a different path from that shown in Figure 7.3, as shown in Figure 7.6. For comparison purposes, the base case results have again reported as SO Step 0.

The reason for this is that the optimizer 'learns' as is progresses. As more design points are evaluated within the design space these improve the linear approximate models used during the LO stage to guide the optimizer towards the optimum solution leading to faster convergence of the algorithm. The optimization shown in Figure 7.3 only has knowledge of the base case result, plus 6 other designs built around the base case to provide a minimum number of design points to define initial trust region for the number of design parameters. The optimization reported in Figure 7.6 has knowledge of the 30 ES designs, plus all design points solved earlier in the GO. Comparing Figures 7.3 and 7.6 shows the benefit of this additional knowledge, as the optimizer finds a better local optimum in fewer steps. This provides further evidence of local minima within the design space.

The global optimal design found is shown in Figures 7.7 and 7.8. The principal flow direction is indicated, being normal to the direction of gravity. Note that this is not guaranteed to be the optimal design within the design space. This optimal design has relatively few, tall, widely spaced fins. The high aspect ratio of the fins clearly excludes extrusion as a fabrication process. This design, and the sub-optimal designs found by local optimization from the base case and the best design in the ES all have fin thickness at or close to the minimum value of 0.5mm, i.e. on the boundary of the design space. This indicates that the heat sink fabrication method needs to be reconsidered.

One option would be to explore designs that are fabricable by extrusion. This could

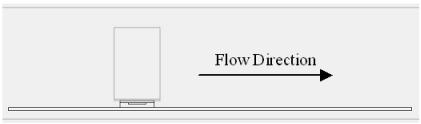

Figure 7.7: Optimal design (a).

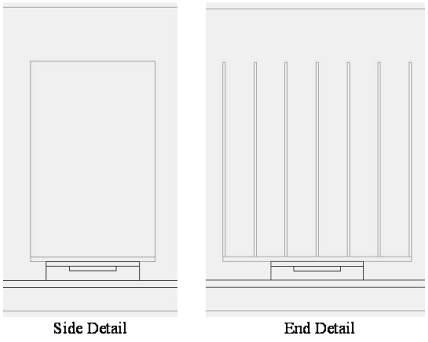

Figure 7.8: Optimal design (b).

be achieved by imposing a constraint on the design variables, so that the fin aspect ratio is less than (say) 10:1:

$$Fin\ Thickness \geq 0.1 * Fin\ Height$$

In addition, the maximum number of fins could be reduced to make the design space smaller and therefore less costly to traverse, as high fin count designs appear to be uninteresting for this natural convection environment.

Another option would be to explore alternative fabrication methods such as folded fin designs, by considering fin thicknesses say down to 0.1mm, without the above constraint and without reducing the maximum number of fins. By exploring both options, the most cost effective solution could be found.

To pursue these options a subsequent ES step should be performed on the new design space. The space-filling characteristics of the LHD technique fit new design points around those already in the feasible design space.

How rapidly the GO finds the best design is a measure of the efficiency of the approach, and is important since it may well be impractical to perform a LO from every design in the ES. In this example, there is a moderate tendency for designs with the lowest objective value (heat sink mass) to result in higher component temperature rises, as shown in Figure 7.9. Despite this, the GO methodology is seen to be very efficient, reducing to within a few percent of the final optimum within just 6 SO steps.

Given the strong evidence for a large number of local minima within the design space it is clear that a more comprehensive ES would have been beneficial, with say 10 or 15 designs per design parameter.

7.8 Conclusions

The use of a simulation-based design optimization methodology has been applied to the optimization of a heat sink design for a given application-specific environment, and the

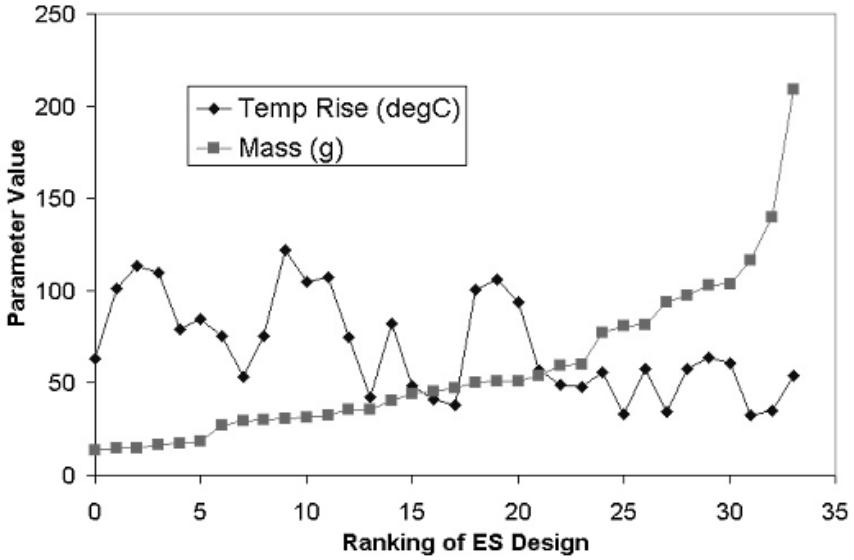

Figure 7.9: ES designs ranked by objective value.

merits of using a global optimization strategy have been shown. The global optimization approach, and the trust region approach used in the local optimization step makes the technique very robust, leading to high confidence that where design improvements are possible, they are achieved.

Heat sink selection is important from a business perspective, as a reduction in heat sink mass can represent a significant overall cost saving in high volume products.

Space constraints increasingly require heat sink designs to be tailored to their environment, and should therefore be optimized as part of the system design. To make the problem more tractable, localized mesh regions can be used to concentrate mesh around the heat sink. Much of the effort required to converge the solution for a new design results from the cell imbalances introduced when the mesh is changed and the dependent variables (pressure, velocities, temperature etc.) are interpolated onto the new mesh. Restricting the mesh changes to a localized region around the heat sink minimizes this.

Once the designs have been created for the ES, each design can be run independently and therefore concurrently, allowing spare capacity on the network to be utilized prior to the LO stage.

At the system-level there are many other design problems that would benefit from automated optimization, such as fan selection and fan positioning, vent sizing and vent positioning, card slot spacing within a sub-rack, etc. The optimization methodology reported here is equally applicable to all levels of packaging, from optimizing the thermal

performance of a chip package for a given cooling strategy, to optimizing the ventilation system and equipment layout in a data center.

Bibliography

E.H.L. Aarts, J.H.M. Korst, and P.J.M. Van Laarhoven (1997). Simulated annealing. In *Local Search in Combinatorial Optimization*, edited by E.H.L. Aarts and J.K. Lenstra, 91–120. John Wiley & Sons, New York.

S.J. Abspoel, L.F.P. Etman, J. Vervoort, R.A. Van Rooij, A.J.G. Schoofs, and J.E. Rooda (2001). Simulation based optimization of stochastic systems with integer design variables by sequential multipoint linear approximation. *Structural and Multidisciplinary Optimization*, 22(2):125–138.

N. Alexandrov, J.E. Dennis, R.M. Lewis, and V. Torczon (1998). A trust region framework for managing the use of approximation models in optimization. *Structural and Multidisciplinary Optimization*, 15(1):16–23.

M.E. Angün (2004). *Black box simulation optimization: generalized response surface methodology*. PhD thesis, Tilburg University.

A.F. Ashour, L.F. Alvarez, and V.V. Toropov (2003). Empirical modelling of shear strength of RC deep beams by genetic programming. *Computers & Structures*, 81(5):331–338.

C. Audet, J.E. Dennis, D.W. Moore, A. Booker, and P.D. Frank (2000). A surrogate-model-based method for constrained optimization. In *Proceedings of the eighth AAIA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*.

M.H. Bakr, J.W. Bandler, R.M. Biernacki, S.H. Chen, and K. Madsen (1998). A trust region aggressive space mapping algorithm for EM optimization. *IEEE Transactions on Microwave Theory and Techniques*, 46(12):2412–2425.

V. Balabanov, A.A. Giunta, O. Golovidov, B. Grossman, W.H. Mason, L.T. Watson, and R.T. Haftka (1999). Reasonable design space approach to response surface approximation. *Journal of Aircraft*, 36(1):308–315.

J.W. Bandler, R.M. Biernacki, S.H. Chen, R.H. Hemmers, and K. Madsen (1995). Electromagnetic optimization exploiting aggressive space mapping. *IEEE Transactions on Microwave Theory and Techniques*, 43(12):2874–2882.

H.J. Bergveld (2001). *Battery management systems - design by modelling*. PhD thesis, Technische Universiteit Twente.

C.M. Bishop (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford.

M. Björkman and K. Holmström (2000). Global optimization of costly nonconvex functions using radial basis functions. *Optimization and Engineering*, 1:373–397.

J.R. Blum (1954). Multidimensional stochastic approximation methods. *Annals of Mathematical Statistics*, 25:737–744.

A.J. Booker, J.E. Dennis, P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset (1999). A rigorous framework for optimization of expensive functions by surrogates. *Structural and Multidisciplinary Optimization*, 17(1):1–13.

G.E.P. Box, W.G. Hunter, and J.S. Hunter (1987). *Statistics for Experimenters*. John Wiley & Sons, New York.

M.J. Box and N.R. Draper (1971). Factorial designs, the $\|X^+X\|$ criterion, and some related matters. *Technometrics*, 13(4):731–742.

R.C.M. Brekelmans, L.T. Driessen, H.J.M. Hamers, and D. Den Hertog (2001). A new sequential optimization approach to product and process design involving expensive simulations. In *Proceedings of the third ASMO UK/ISSMO conference on Engineering Design Optimization: Product and Process Improvement*, edited by Osvaldo M. Querin, 49–52.

R.C.M. Brekelmans, L.T. Driessen, H.J.M. Hamers, and D. Den Hertog (2005). Constrained optimization involving expensive function evaluations: a sequential approach. *European Journal of Operational Research*, 160(1):121–138.

M. Bremicker, P.Y. Papalambros, and H.T. Loh (1990). Solution of mixed-discrete structural optimization problems with a new sequential linearization algorithm. *Computers & Structures*, 37(4):451–461.

A.R. Conn, N.I.M. Gould, and Ph.L. Toint (2000). *Trust-Region Methods*. MPS/SIAM series on optimization. SIAM, Philadelphia.

A.R. Conn, K. Scheinberg, and Ph.L. Toint (1997). Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming*, 79(3):397–414.

A.R. Conn, K. Scheinberg, and L.N. Vicente (2004). Geometry of sample sets in derivative free optimization, part I: polynomial interpolation. *Technical report*.

A.R. Conn, K. Scheinberg, and L.N. Vicente (2005). Geometry of sample sets in derivative free optimization, part II: polynomial regression and underdetermined interpolation. *Technical report*.

A.R. Conn and Ph.L. Toint (1996). An algorithm using quadratic interpolation for unconstrained derivative free optimization. In *Nonlinear Optimization and Applications*, 27–47. Plenum, New York.

K.J. Craig and N. Stander (2002). An improved version of DYNAMIC_Q for simulation-based optimization using response surface gradients and an adaptive trust region. *European Journal of Operational Research*, 140(2):197–211.

A.A.M. Cuyt, R.B. Lenin, S. Becuwe, and B.M. Verdonk (2004). Adaptive multivariate rational data fitting with applications in electromagnetics. *Technical report*, University of Antwerp.

P.J. Davis (1975). *Interpolation and Approximation*. Dover Publications, New York.

D. Den Hertog and H.P. Stehouwer (2002a). Optimizing color picture tubes by high-cost non-linear programming. *European Journal of Operational Research*, 140(2):197–211.

D. Den Hertog and H.P. Stehouwer (2002b). Product and process optimisation with simulation. In *Proceedings of 3rd EuroSimE Conference*, 179–192. Paris.

J.E. Dennis and R. Schnabel (1989). A view of unconstrained optimization. In *Handbook of Operations Research and Management Science, Volume 1, Optimization*, edited by G.L. Nemhauser et al., 1–72. Elsevier Science Publishers B.V., Amsterdam.

J.E. Dennis and V. Torczon (1994). Derivative-free pattern search methods for multidisciplinary design problems. In *Proceedings of the Fifth AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 922–932. MCB University Press, Panama City Beach.

T. Dhaene (2002). Automated fitting and rational modeling algorithm for EM-based S-parameter data. In *Applied Parallel Computing. Advanced Scientific Computing: sixth International Conference, PARA 2002*, edited by J. Fagerholm et al., 99–105. Espoo.

L.C.W. Dixon (1994). On automatic differentiation and continuous optimization. In *Algorithms for Continuous Optimization: the State of the Art*, edited by E. Spedicato, 501–513. Kluwer Academic Publishers, Dordrecht.

J.M. Donohue, E.C. Houck, and R.H. Myers (1993). Simulation designs and correlation induction for reducing second-order bias in first-order response surfaces. *Operations Research*, 41(5):880–902.

J.M. Donohue, E.C. Houck, and R.H. Myers (1995). Simulation designs for the estimation of quadratic response surface gradients in the presence of model misspecification. *Management Science*, 41(2):244–262.

L.T. Driessen, R.C.M Brekelmans, M. Gerichhausen, H.J.M Hamers, and D. Den Hertog (2005). Why methods for optimization problems with time-consuming function evaluations and integer variables should use global approximation models. *Center discussion paper 2006-04*, Tilburg University.

L.T. Driessen, R.C.M Brekelmans, H.J.M Hamers, and D. Den Hertog (2006). On D-optimality based trust regions for black-box optimization problems. *Structural and Multidisciplinary Optimization*, 31(1):40–48.

O. Dykstra (1971). The augmentation of experimental data to maximize $|X'X|$. *Technometrics*, 13:682–688.

J. Dyson (1993). Laminar and turbulent flow and heat transfer between parallel surfaces. *Document flotherm/cs1/0293 issue 1.0*, Flomerics Ltd.

Y.M. Ermoliev (1988). Stochastic quasigradients methods. In *Numerical Techniques for Stochastic Optimization*, edited by Y. Ermoliev and R.J.B. Wets. Springer-Verlag, Berlin.

L.F.P. Etman (1997). *Optimization of multibody systems using approximation concepts*. PhD thesis, Eindhoven University of Technology.

R. Fletcher and S. Leyffer (2002). Nonlinear programming without a penalty function. *Mathematical Programming*, 91:239–270.

J.A.A. Gijsbers (2003). Integer simulation-based optimization: a new multipoint approximation approach with probabilistic filter acceptance and intermediate design variables. *Systems engineering report se-420398*, Eindhoven University of Technology.

P.E. Gill, W. Murray, and M.H. Wright (1981). *Practical Optimization*. Academic Press, New York.

F.W. Glover, J.P. Kelly, and M. Laguna (1996). New advances and applications of combining simulation and optimization. In *Proceedings of the 1996 Winter Simulation Conference*, edited by J. Charnes et al., 144–152. Coronado.

F.W. Glover, J.P. Kelly, and M. Laguna (1999). New advances for wedding optimization and simulation. In *Proceedings of the 1999 Winter Simulation Conference*, edited by P.H. Farrington et al., 255–260. Phoenix.

F.W. Glover and M. Laguna (1997). *Tabu Search*. Kluwer Academic Publishers, Norwell.

P.W. Glynn (1989). Optimization of stochastic systems via simulation. In *Proceedings of the 1989 Winter Simulation Conference*, edited by E.A. MacNair et al., 90–105. Washington.

G.H. Golub and C.F. Van Loan (1996). *Matrix Computations*. The John Hopkins University Press, Baltimore, 3rd edition.

A. Griewank (1989). On automatic differentiation. In *Mathematical Programming: Recent Developments and Applications*, edited by M. Iri and K. Tanabe, 83–108. Kluwer, Dordrecht.

H.-M. Gutmann (2001). A radial basis function method for global optimization. *Journal of Global Optimization*, 19(3):201–227.

J.H. Holland (1975). *Adaptation in Natural and Artificial Systems*. M.I.T. Press, Cambridge, 1st edition.

R. Hooke and T.A. Jeeves (1961). Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery*, 8(2):212–229.

B.G.M. Husslage, E.R. Van Dam, D. Den Hertog, H.P. Stehouwer, and E.D. Stinstra (2003). Collaborative metamodeling: coordinating simulation-based product design. *Concurrent Engineering: Research and Applications*, 11(4):267–278.

J.H. Jacobs (2004). *Performance quantification and simulation optimization of manufacturing flow lines*. PhD thesis, Eindhoven University of Technology.

D.R. Jones (2001a). DIRECT global optimization algorithm. In *Encyclopedia of Optimization*, edited by C.A. Floudas and P.M. Pardalos, volume 1, 431–440. Kluwer Academic Publishers, Dordrecht.

D.R. Jones (2001b). A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383.

D.R. Jones, C.D. Perttunen, and B.E. Stuckman (1993). Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181.

D.R. Jones, M. Schonlau, and W.J. Welch (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492.

J. Kiefer and J. Wolfowitz (1952). Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23(3):462–466.

J.P.C. Kleijnen, D. Den Hertog, and M.E. Angün (2004). Response surface methodology's steepest ascent and stepsize revisited. *European Journal of Operational Research*, 159(1):121–131.

J.P.C. Kleijnen and W.C.M. Van Beers (2004). Application-driven sequential designs for simulation experiments: Kriging metamodelling. *Journal of the Operational Research Society*, 55(9):876–883.

J.R. Koehler and A.B. Owen (1996). Computer experiments. In *Handbook of Statistics*, edited by S. Ghosh and C.R. Rao, volume 13, 261–308. Elsevier Science B.V., Boston.

J.R. Koza (1994). Genetic programming as a means for programming computers by natural-selection. *Statistics and Computing*, 4(2):87–112.

M. Laguna and R. Martí (2002). Neural network prediction in a system for optimizing simulations. *IIE Transactions*, 34(3):273–282.

P. L'Ecuyer (1991). An overview of derivative estimation. In *Proceedings of the 1991 Winter Simulation Conference*, edited by B.L. Nelson et al., 207–217. IEEE Press, Phoenix.

P. L'Ecuyer and G. Perron (1994). On the convergence rates of IPA and FDC derivative estimators for finite-horizon stochastic systems. *Operations Research*, 42(4):643–656.

R.M. Lewis, V. Torczon, and M.W. Trosset (2001). Direct search methods: then and now. In *Numerical Analysis 2000*, volume 4, 191–207. Elsevier, North Holland.

S. Leyffer (2001a). Generalized outer approximation. In *Encyclopedia of Optimization*, edited by C.A. Floudas and P.M. Pardalos, volume 2, 247–254. Kluwer Academic Publishers, Dordrecht.

S. Leyffer (2001b). Integrating SQP and branch-and-bound for mixed integer nonlinear programming. *Computational Optimization & Applications*, 18:295–309.

H.T. Loh and P.Y. Papalambros (1991a). Computational implementation and tests of a sequential linearization algorithm for mixed-discrete nonlinear design optimization. *Journal of Mechanical Design*, 113(3):335–345.

H.T. Loh and P.Y. Papalambros (1991b). A sequential linearization approach for solving mixed-discrete nonlinear design optimization problems. *Journal of Mechanical Design*, 113(3):325–334.

M. Marazzi and J. Nocedal (2002). Wedge trust region methods for derivative free optimization. *Mathematical Programming*, 91(2):28–306.

T.J. Mitchell (1974). An algorithm for the construction of d-optimal experimental designs. *Technometrics*, 16(4):203–210.

D.C. Montgomery (1984). *Design and Analysis of Experiments*. John Wiley & Sons, New York, 2nd edition.

M.D. Morris and T.J. Mitchell (1995). Exploratory designs for computer experiments. *Journal of Statistical Planning and Inference*, 43:381–402.

R.H. Myers and D.C. Montgomery (1995). *Response Ssurface Methodology: Process and Product Optimization using Designed Experiments*. John Wiley & Sons, New York.

J. Parry, R.B. Bornoff, H.P. Stehouwer, L.T. Driessen, and E.D. Stinstra (2004). Simulation-based design optimization methodologies applied to CFD. *IEEE Transactions on Components and Packaging Technologies*, 27(2):391–397.

M.J.D. Powell (1994a). A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in Optimization and*

Numerical Analysis, Proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico, edited by S. Gomez and J.P. Hennart, volume 275, 5–67. Kluwer Academic Publishers, Dordrecht.

M.J.D. Powell (1994b). A direct search optimization method that models the objective by quadratic interpolation. In *Presentation at the 5th Stockholm Optimization Days*.

M.J.D. Powell (2001). Radial basis function methods for interpolation to functions of many variables. *DAMTP 2001/NA11*, University of Cambridge.

M.J.D. Powell (2002). UOBYQA: unconstrained optimization by quadratic approximation. *Mathematical Programming*, 92(3):555–582.

M.J.D. Powell (2003). Least Frobenius norm updating of quadratic models that satisfy interpolation conditions. *Mathematical Programming*, 100(1):183–215.

M.J.D. Powell (2004). On the use of quadratic models in unconstrained minimization without derivatives. *Optimization Methods & Software*, 19(3-4):399–411.

R.G. Regis and C.A. Shoemaker (2005). Constrained global optimization of expensive black box functions using radial basis functions. *Journal of Global Optimization*, 31(1):153–171.

J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn (1989). Design and analysis of computer experiments. *Statistical Science*, 4:409–435.

M.H. Safizadeh (2002). Minimizing the bias and variance of the gradient estimate in RSM simulation studies. *European Journal of Operational Research*, 136:121–135.

T.J. Santner, B.J. Williams, and W.I. Notz (2003). *The design and analysis of computer experiments*. Springer-Verlag, New York.

K. Schittkowski (1987). *More test examples for nonlinear programming codes*, volume 282 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin.

A.J.G. Schoofs, P.J.M. Roozen-Kroon, and D.H. Van Campen (1994). Optimization of structural and acoustical parameters of bells. In *Proceedings of the Fifth AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. MCB University Press, Panama City Beach.

C. Shen, G.-J. Cao, and X.-J. Zhu (2002). Nonlinear modeling of MCFC stack based on RBF neural networks identification. *Simulation Modelling Practice and Theory*, 10(1-2):109–119.

A. Sóbester, S.J. Leary, and A.J. Keane (2004). A parallel updating scheme for approximating and optimizing high fidelity computer simulations. *Structural and Multidisciplinary Optimization*, 27(5):371–383.

H.P. Stehouwer and D. Den Hertog (1999). Simulation-based design optimisation: methodology and applications. In *Proceedings of the First ASMO UK / ISSMO Conference on Engineering Design Optimization*, 349–355. MCB University Press, Ilkley.

E.D. Stinstra, L.T. Driessen, and H.P. Stehouwer (2001). Design optimization: some pitfalls and their remedies. In *Proceedings of the third ASMO UK/ISSMO conference on Engineering Design Optimization: Product and Process Improvement*, edited by Osvaldo M. Querin, 203–210.

V. Torczon (1992). PDS: Direct search methods for unconstrained optimization on either sequential or parallel machines. *Technical report 92-09*, Rice University, Department of Computational and Applied Mathematics.

V.V. Toropov (1999). Multipoint approximations based on response surface fitting: a summary of recent developments. In *Proceedings of the First ASMO UK / ISSMO Conference on Engineering Design Optimization*, 371–380. MCB University Press, Ilkley.

V.V. Toropov, A.A. Filatov, and A.A. Polynkin (1993). Multiparameter structural optimization using FEM and multipoint explicit approximations. *Structural and Multidisciplinary Optimization*, 6:7–14.

W.C.M. Van Beers and J.P.C. Kleijnen (2004). Kriging interpolation in simulation: a survey. In *Proceedings of the 2004 Winter Simulation Conference*, edited by R. Ingalls et al., 113–121. Piscataway.

F. Van Keulen and K. Vervenne (2002). Gradient-enhanced response surface building. In *Proceedings of the Ninth AIAA / ISSMO Symposium and Exhibit on Multidisciplinary Analysis and Optimization*. Atlanta.

F. Vanden Berghen and H. Bersini (2005). CONDOR, a new parallel, constrained extension of Powell's UOBYQA algorithm: experimental results and comparison with the DFO algorithm. *Journal of Computational and Applied Mathematics*, 181(1):157–175.

Y. Ye (1992). A new complexity result on minimization of a quadratic function over a sphere constraint. In *Recent Advances in Global Optimization*, edited by C.A. Floudas and P.M. Pardalos, 19–21. Princeton University Press, Princeton.

M.A. Zazanis and R. Suri (1993). Convergence rates of finite-difference sensitivity estimates for stochastic systems. *Operations Research*, 41(4):694–703.

Samenvatting

Dit proefschrift behandelt optimalisatie rondom simulatiemodellen. Het bestaat uit een zestal artikelen, voorafgegaan door een inleidend hoofdstuk. Dit inleidend hoofdstuk bespreekt de ontwikkelingen op het gebied van optimalisatie rondom computer simulaties, definieert het bijbehorend wiskundig optimalisatieprobleem, en gaat in op de verschillende optimalisatiemethoden voor dit soort problemen.

Het ontwerpen van produkten en processen is langzamerhand verschoven van een puur fysiek proces naar een proces dat ondersteund wordt door computersimulaties en virtuele prototypes. Om dit virtuele ontwerpproces te optimaliseren in termen van snelheid en produktkwaliteit worden steeds vaker statistische technieken en wiskundige optimalisatiemethoden toegepast. Het resulterend wiskundig optimalisatieprobleem is niet standaard in de zin dat zowel in de doelfunctie als in eventuele restricties, afhankelijke variabelen kunnen voorkomen die niet expliciet zijn. Deze variabelen kunnen alleen geëvalueerd worden middels een computersimulatie. De tijdsduur van een enkele computersimulatie kan uiteenlopen van minder dan enkele seconden tot een hele dag.

Afhankelijk van deze simulatieduur, kan het optimalisatieprobleem op verschillende manieren opgelost worden. Als één simulatie minder dan enkele seconden in beslag neemt, kunnen klassieke optimalisatiemethoden gebruikt worden. Voor optimalisatie rondom tijdrovende simulaties bestaan specifieke methoden, die de tijdrovende simulatietool vervangen door snel evalueerbare benaderende functies. Deze methoden kunnen worden onderverdeeld in doelfunctie-gedreven methoden en model-gedreven methoden. In doelfunctie-gedreven methoden wordt de selectie van simulatiekandidaten vooral bepaald door hun voorspelde verbetering van de doelfunctie. Model-gedreven methoden selecteren kandidaten op basis van hun verwachte invloed op de nauwkeurigheid van de benaderende functies. De focus ligt in dit proefschrift, wat betreft de specifieke methoden voor tijdrovende simulaties, vooral op de doelfunctie-gedreven methoden. Voor deze klasse wordt een algemeen raamwerk gegeven en elke stap in dit raamwerk wordt besproken aan de hand van relevante literatuur op dit gebied. Binnen dit raamwerk maken we onderscheid tussen methoden die gebaseerd zijn op lokale benaderende functies en methoden die gebaseerd zijn op globale benaderende functies.

De eerste bijdrage van dit proefschrift ligt op het gebied van gradiëntenbenaderingen voor simulatiemodellen die in maximaal een paar seconden doorgerekend kunnen worden. Vaak worden hier klassieke optimalisatiemethoden, zoals SQP en GRG, voor gebruikt. In Hoofdstuk 2 en 3 analyseren we een belangrijke stap binnen elke klassieke optimalisatiemethode, namelijk het schatten van de gradiënten. Een veel gebruikte techniek hiervoor is 'finite differencing'. Het succes van de optimalisatiemethode wordt mede bepaald door de nauwkeurigheid van de gradiëntschattingen. Dit geldt in het bijzonder voor simulatieproblemen, vanwege de mogelijke invloed van numerieke of stochastische ruis.

Hoofdstuk 2 analyseert en vergelijkt verschillende schema's om gradiëntschattingen te verkrijgen voor onderliggende functies die aan ruis onderhevig zijn. Als foutcriterium wordt de gemiddelde kwadratische fout (MSE) gebruikt. Voor drie finite differencing schema's en twee schema's uit de 'Design of Experiments' analyseren we de stochastische en de deterministische fout. We leiden voor elk schema de optimale stapgrootte af en tonen de toepasbaarheid van de Design of Experiments schema's aan in een toepassing op financieel gebied. We concluderen dat voor functies die onderhevig zijn aan ruis, statistische proefopzetten goed voldoen als methode om gradiënten te schatten.

Gradiëntschattingen kunnen ook verkregen worden door toepassing van interpolerende Lagrange polynomen. Hoofdstuk 3 analyseert de MSE bij het gebruik van (op N herhalingen gebaseerde) interpolerende Lagrange polynomen. We laten zien dat de gemiddelde kwadratische fout van de orde $N^{-1+\frac{1}{2d}}$ is, als we de schattingsprocedure N keer herhalen en $2d$ evaluatiepunten per herhaling gebruiken. Hieruit volgt dat de orde van de MSE naar N^{-1} convergeert als het aantal evaluatiepunten naar oneindig gaat. We bepalen de optimale verhouding tussen aantal evaluatiepunten en aantal herhalingen voor ieder gegeven totaal aantal evaluaties. Verder tonen we aan dat de schattingen van de gradiënten robuuster zijn naarmate er meer evaluatiepunten gebruikt worden. Testresultaten laten zien dat voor lage ruisniveau's Lagrange polynomen beter presteren dan central finite differencing. Voor hogere ruisniveau's komt de Lagrange methode neer op central finite differencing.

De tweede bijdrage ligt op het gebied van de doelfunctie-gedreven methoden voor optimalisatie rondom tijdrovende simulaties. Hoofdstuk 4 presenteert een nieuwe doelfunctie-gedreven methode, genaamd SEQUEM, die gebaseerd is op lokale benaderende functies. Gebruik makend van de filter methode van Fletcher en Leyffer, kunnen we problemen met allerlei randvoorwaarden optimaliseren zonder toepassing van een penaltyfunctie. Het filterconcept levert de ontwerper ook een set met interessante ontwerpen op, in plaats van een enkel optimaal ontwerp. De kwaliteit van de lokale benaderingen wordt bewaakt door toepassing van een geometriemaat voor de locatie van de simulatiepunten. Een

aantal van de in dit hoofdstuk beschreven technieken zijn succesvol geïmplementeerd in RMS-Opt, een doelfunctie-gedreven methode die al jaren intensief gebruikt wordt binnen LG.Philips Displays.

Hoofdstuk 5 analyseert twee belangrijke bouwstenen van op lokale benaderingen gebaseerde doelfunctie-gedreven methoden: de zogenaamde 'trust region' en het mechanisme om een goede geometrie te bewerkstelligen van de punten waarop de benaderende functies gebaseerd zijn. We stellen voor om het concept D-optimaliteit, dat bekend is in de wereld van statistische proefopzetten, te gebruiken in de definitie van de trust region en als geometriemaat. Resultaat is een ellipsoïde trust region, waarvan de vorm en lokatie volledig bepaald worden door de punten waarop de benaderingen gebaseerd zijn. Verder tonen we aan dat de ellipsoïde trust region onafhankelijk is van affiene transformaties. De voorgestelde trust region en geometriemaat kunnen beiden geïntegreerd worden in bestaande op lokale benaderingen gebaseerde doelfunctie-gedreven methoden.

In recente publicaties worden optimalisatieproblemen met tijdrovende simulaties en integer ontwerpvariabelen opgelost met doelfunctie-gedreven methoden gebaseerd op *lokale* benaderende functies. In Hoofdstuk 6 bepleiten we het gebruik van methoden die gebaseerd zijn op *globale* benaderende functies voor dit soort problemen. We laten zien dat methoden die gebaseerd zijn op lokale benaderende functies kunnen leiden tot de afronding van de continue optimale oplossing, of tot nog slechtere oplossingen. We stellen een methode voor die gebaseerd is op globale benaderende functies. Testresultaten laten zien dat deze methode goed presteert, zowel voor analytische als practische testproblemen.

Hoofdstuk 7 beschrijft de succesvolle toepassing van een doelfunctie-gedreven methode bij het ontwerp van een afkoelklem (die bijvoorbeeld te vinden is bovenop een CPU van een computer). De simulaties zijn in dit geval erg tijdrovend. De resultaten illustreren de doeltreffendheid van de gebruikte methode.